Lauwerier – Spelen met GRAPHICS EN FRACTALS

Spelen met GRAPHICS EN FRACTALS

80 programma's in Q(uick)BASIC en PowerBASIC

Hans Lauwerier

B

Spelen met graphics en fractals

In de serie Spelen met ... verschijnen:

Spelen met the 7th Guest
Spelen met Microsoft Flight Simulator 5
Spelen met Microsoft Space Simulator
Spelen met SimCity 2000
Spelen met Virtual Reality
Spelen met graphics en fractals
Spelen met geluid op de PC
Spelen met fractal graphics voor Windows
Spelen met MYST

P

Hans Lauwerier

SPELEN MET GRAPHICS EN FRACTALS

80 programma's
in Q(uick)BASIC
en PowerBASIC

ACADEMIC SERVICE
informatica

E

CIP-GEGEVENS KONINKLIJKE BIBLIOTHEEK, DEN HAAG

Lauwerier, H.A.

Spelen met graphics en fractals : 80 programma's in
Q(uick)BASIC en PowerBASIC / H. Lauwerier. - Schoonhoven : Academic
Service. - Ill. + diskette
Met lit. opg., reg.
ISBN 90-395-0092-4
NUGI 852
Trefw.: fractals / QuickBasic (programmeertaal).

Uitgegeven door: Academic Service, Schoonhoven
Zetwerk: Redactiebureau R. Heyer, Markelo
Omslagontwerp: Studio Cursief, Amsterdam
Druk omslag: Casparie, IJsselstein
Druk: Krips Repro Meppel
Bindwerk: Meeuwis, Amsterdam

ISBN 90 395 0092 4
NUGI 852

F

Inhoud

G

H

Inleiding

De moderne huiscomputer, een MS-DOS machine met een 80386- of hogere processor, heeft onvoorstelbare mogelijkheden op het gebied van rekenen en tekenen. Dankzij de alom verkrijgbare software zijn tekstverwerking en administratie betrekkelijk eenvoudig geworden. Zonder veel moeite kunnen allerlei ingewikkelde documenten vervaardigd worden. Grafische pakketten maken het mogelijk om documenten van plaatjes te voorzien. Al dit moois kan bereikt worden zonder kennis te hebben van de hogere programmeertalen. Wie echter de moeite genomen heeft vertrouwd te raken met bijvoorbeeld Basic, de beginnerstaal van elke computerbezitter, wordt ruimschoots beloond. Niet alleen geeft het een grote voldoening zelf een programma te schrijven dat werkt, maar ook heeft men de mogelijkheid iets geheel nieuws te maken, figuren te ontwerpen die de indruk van computerkunst kunnen geven. Weliswaar is voor het programmeren enige wiskundige kennis nodig, maar die kennis hoeft niet groter te zijn dan wat men in een paar klassen VWO opsteekt. Het belangrijkste is vertrouwdheid met eenvoudige wiskundige functies zoals de sinus-, de logaritme- en de exponentiële functie. Voorts is enige ervaring in het gebruik van de coördinatenmeetkunde vereist.

In de computerwereld wordt vaak gebruik gemaakt van een of andere Basic-versie. Ooit was Basic een 'beginnerstaaltje' waarvoor experts hun neus optrokken. De universiteiten propageerden onder andere Pascal als een veel betere taal die in tegenstelling tot Basic tot beter gestructureerd programmeren noopte. Oorspronkelijk kon men bij Pascal gebruik maken van een zogenaamde compiler, een vertaalprogramma dat het oorspronkelijk in Pascal geschreven programma in zijn totaliteit in machinetaal omzette. Bij Basic was dat aanvankelijk niet het geval en dat was ook de reden dat in Basic geschreven programma's vroeger berucht waren om hun trage werking. Inmiddels zijn veel jaren verstreken, de computertalen zijn naar elkaar toegegroeid en voorzien van voorvoegsels als Turbo, Quick, enzovoort. Nog steeds is Basic een vrij gemakkelijk te leren taal maar nu zijn er goede compilers beschikbaar. De aandacht voor Pascal is enigszins teruggelopen ten gunste van de programmeertaal C, een echte taal voor professionals. Basic in de vorm van moderne compilerversies als Turbo Basic, QuickBasic en Power Basic heeft terrein herwonnen en kan nu beschouwd worden als een volwassen programmeertaal waarin men goed leesbare en goed gestructureerde programma's kan schrijven. In 1993 is van Power Basic de nieuwe versie 3.0 verschenen met nog meer mogelijkheden, in het bijzonder ook op

grafisch gebied. Tegenwoordig wordt bij MS-DOS standaard de programmeertaal QBasic meegeleverd. Dit is een soort uitgeklede versie van QuickBasic, het heeft geen compiler en is dan ook interpretatief. Dat wil zeggen dat bij QBasic de opdrachten tijdens de programma-uitvoering regel voor regel in machinetaal worden omgezet. Dat kan bij programma's met veel lussen tot een aanzienlijke vertraging leiden.

Basic is vooral geschikt voor het schrijven van kleine programma's, programma's van enige tientallen regels die royaal op een A4'tje passen. Die programma's kunnen zelfstandig als losse eenheden gebruikt worden zonder dat het nodig is hulpprogramma's vanuit een 'library' aan te roepen. De voordelen van Basic springen vooral in het oog bij het gebruik van grafische opdrachten. Helaas komen de grafische aspecten van het programmeren er in veel handleidingen bekaaid af. Dit boek laat echter zien wat u met kleine grafische programma's kunt bereiken. Alle te bespreken programma's hebben iets te maken met het maken van een figuur of een plaatje waaraan men begrippen als leuk, mooi, interessant of fraai kan hechten. Sommige programma's illustreren bepaalde natuurkundige of wiskundige eigenschappen. Wie niet zo bekend is met die achtergrond zal alleen maar mooie of interessante plaatjes zien, maar deskundigen kunnen er extra plezier aan beleven. De programma's zijn ook bedoeld als modelprogramma's die men naar eigen willekeur kan veranderen.

Veel programma's zouden best verbeterd of misschien verkort kunnen worden. Iedereen die programma's schrijft, weet dat de eisen van duidelijkheid en kortheid met elkaar in conflict zijn. Om typografische redenen is weinig toelichtende tekst in de programma's opgenomen. Ook worden constanten en variabelen meestal met slechts een of twee symbolen aangeduid. In de toelichting bij elk programma wordt de werking van het programma echter uitvoerig toegelicht. Wie de programma's voor eigen gebruik overneemt, kan natuurlijk alles naar eigen inzicht benoemen.

Sommige programma's bevatten een aantal grootheden, de zogenaamde parameters, die door de gebruiker (een beetje) gewijzigd kunnen worden. Daartoe hoeft men vaak slechts een enkele opdrachtregel te wijzigen, een proces van 'editing'. Het loont de moeite daarmee vertrouwd te raken. Op die manier kan een enkel programma een grote verscheidenheid van resultaten opleveren.

De programma's in dit boek zijn geordend naar onderwerp. Tevens is een enigszins vage lijn van eenvoudig naar moeilijk aangehouden. In de eerste twee hoofdstukken worden naast wat eenvoudige wiskundige theorie de eenvoudigste principes van de grafische bewerkingen en het

hanteren van kleuren behandeld. De eerste programma's bestaan uit slechts enkele regels, het laatste programma MZOOML beslaat drie A4'tjes, maar dit programma biedt dan ook uitgebreide mogelijkheden. Het stelt de gebruiker in staat op interactieve wijze willekeurig kleine details van de bekende Mandelbrot set tot beeldschermvullende kleurige beelden op te blazen.

1 Programmeren in graphics

1.1 Inleiding

Bij het schrijven van programma's in een moderne versie van Basic moet een aantal zaken in het oog worden gehouden. Hoewel de verschillende Basic-dialecten onderling weinig verschillen, kan men soms voor verrassingen komen te staan. Dat betreft vooral het gebruik van kleuren in bijvoorbeeld de VGA-modus bij Q(uick)Basic en Power Basic. Daarnaast zijn er kleine, maar hinderlijke verschillen wanneer men bijvoorbeeld een plaatje van tekst wil voorzien of wanneer men in een programma een kleine pauze wil inlassen.

In zijn oorspronkelijke vorm was de PC alleen voor schrijvers en boekhouders bedoeld. Het beeldscherm, meestal rechthoekig in de verhouding 4:3, werd beschouwd als een vel papier met een breedte van 40 of 80 kolommen (tekens) en een lengte van 25 regels, te tellen van boven naar beneden. De letters werden opgebouwd uit beeldpuntjes, de zogenaamde pixels, waarvan er op het scherm 320*200 beschikbaar waren. Inmiddels heeft de elektronica een enorme ontwikkeling doorgemaakt en beschikken de meeste moderne huiscomputers over grafische kaarten met geavanceerde mogelijkheden. Voor grafische programma's komt daarbij eigenlijk alleen de VGA-modus (of nog beter) in aanmerking met een resolutie van 640*480 pixels en 16 kleuren. Omdat 640:480 overeenkomt met 4:3, zijn de pixels daarbij als een onzichtbaar ruitjesgordijn over het scherm verspreid met gelijke onderlinge afstanden in zowel de horizontale als de verticale richting. Dat laatste is van belang wanneer u bijvoorbeeld vierkanten en cirkels wilt afbeelden. Een VGA-kaart is neerwaarts compatible. Dit wil zeggen dat men desgewenst ook andere modi, gebruikt bij vroeger geschreven software, kan toepassen. De VGA-modus is eigenlijk een vervanger of opvolger van de EGA-modus. Het lijkt er op dat VGA op zijn beurt opgevolgd gaat worden door

SVGA-modi met nog hogere resoluties als 800:600. Maar voorlopig ontbreekt de nodige software om daar plezierig mee te kunnen werken.

De EGA-modus kent een oplossend vermogen van 640*350, toch wel redelijk. Een bezwaar bij grafische programma's is echter het verschil in de horizontale en verticale lengte-eenheden. Wie een cirkel denkt te programmeren, kan in de EGA-modus een ellips verwachten, vierkanten worden platte rechthoeken, enzovoort. In Basic wordt de VGA-modus aangeroepen met de opdracht SCREEN 12, terwijl de EGA-modus met het schermgetal 9 correspondeert.

Dit boek is geen handleiding voor Basic, maar wel wordt soms wat meer in detail ingegaan op bepaalde veelgebruikte en nuttige grafische opdrachten. Hierbij gaat het in het bijzonder om het gebruik van kleuren, een onderwerp dat in de gebruikelijke handleidingen weleens te beknopt en zelfs onvolledig wordt behandeld. Gebruikers die niet regelmatig met Basic werken, doen er goed aan een beknopte handleiding, bijvoorbeeld een zakboekje, bij de hand te houden. Uitvoeriger boeken die verderop ook geciteerd worden, zijn:

Basiscursus QBasic, Nok van Veen c.s., Academic Service 1993.
Werken met QBasic, Phil Feldman, Academic Service 1992.
QuickBasic 4.5, Programmeren voor gevorderden, Peter Aitken, Academic Service 1990.

De eerste twee boeken worden gemakshalve als QBB en QBW geciteerd, gevolgd door een aanduiding van het hoofdstuk, de paragraaf of de pagina.

Hoewel dit boek ook geen wiskundige handleiding is, wordt ervan uitgegaan dat de lezer over enige wiskundige basiskennis beschikt. Zo dient u bekend te zijn met het gebruik van de sinus- en cosinus-functie, de logaritme- en de exponentiële functie. Hierbij gaat het niet zozeer om de kennis van allerlei formules – deze kunnen in een tabel of in een wiskundehandleiding opgezocht worden – maar om het vertrouwd zijn met de grafische voorstelling van deze functies. Van belang is verder bekendheid met het gebruik van een x,y-coördinatenstelsel, de afstandformule en de vergelijking van een cirkel.

De programma's in dit boek zijn geschreven in een overzichtelijke, moderne vorm van Basic, dat wil zeggen zonder regelnummers en met inspringen van de tekst bij lusconstructies, enzovoort. Gestreefd is naar duidelijkheid en beknoptheid. De programma's kunnen zonder meer worden gebruikt, maar sommigen zullen er allicht wat meer verklarende en beschrijvende tekst aan willen toevoegen. Andere gebruikers zullen geneigd zijn het programma hier en daar in te korten. Daar is niets op

tegen, elke programmeur is ontevreden over de programma's van vroeger die hij/zij onder ogen krijgt. Het belangrijkste criterium is echter dat een programma begrijpelijk en leesbaar blijft, ook wanneer u er lange tijd niet naar gekeken hebt.

Het verfraaien van een programma is niet altijd lonend, zeker niet wanneer het programma weinig gebruikt zal worden. Als schakel in een groter geheel, een subroutine of een procedure, kan de bereikte tijdwinst heel welkom zijn.

Omdat de programma's meestal vrij kort zijn is afgezien van de zogenaamde modulaire programmeertechniek, zoals aanbevolen in QBB. Dat betekent dat in het algemeen alleen de DEF FN- en GOSUB-constructie worden toegepast.

De programma's in dit boek kan men opvatten als modellen waarnaar men zijn eigen programma's kan inrichten. Om layout-technische redenen is soms een aantal korte opdrachten op dezelfde regel geplaatst. Evenals in de wiskunde worden variabelen en constanten meestal met een of twee symbolen aangeduid. Wie niet opziet tegen extra typewerk, kan daar natuurlijk verandering in aanbrengen. Men kan de programma's gebruiken als uitgangspunt om verder te experimenteren, bijvoorbeeld door een of meer getallen te wijzigen. Het loont de moeite de techniek van 'editing' goed onder de knie te krijgen. Een enkel programma kan soms leiden tot een hele familie van leuke programma's die elk weer iets anders doen.

1.2 Declareren

Aan het begin van een programma kunnen de te gebruiken grootheden gedeclareerd worden als behorende tot een bepaald type. Basic is daar nogal makkelijk in, omdat op een zinvolle wijze van default-typen gebruik gemaakt wordt. In veel programma's wordt gewerkt met integers (gehele getallen) en enkele precisie-getallen (integer % en real !) en dan kan de declaratie achterwege blijven. Bij sommige programma's, bijvoorbeeld de Mandelbrot set, moet er lang en nauwkeurig gerekend worden, en dan is het nodig een aantal variabelen als dubbele precisievariabele te declareren, bijvoorbeeld als DEFDBL S-Z.

Arrays moeten altijd gedeclareerd worden met een DIM-opdracht, bijvoorbeeld DIM A(3). De bedoeling is dat van te voren een aantal adressen voor de array-variabelen wordt gereserveerd. In sommige dialecten is dit onnodig als het aantal variabelen klein is, maar het is beter nauwkeurig te werk te gaan. Het is minder bezwaarlijk wat meer adresruimte

te reserveren dan nodig is dan dat u net een plaatsje tekort komt. Soms is het handig een DIM-opdracht onbepaald te laten, zoals in de volgende combinatie van twee opdrachten:

```
M=8 : DIM X(M),Y(M)
```

Wanneer u hetzelfde programma later nog eens wilt uitvoeren met een andere waarde van *m*, een variabele die ook verderop in het programma kan voorkomen, hoeft u slechts op een enkele plaats de waarde van *m* te wijzigen.

1.3 Integers

In bepaalde grafische programma's moet veel geteld worden. Soms bestaat een tekening uit meer dan een miljoen puntjes. Daarbij kunnen problemen ontstaan als een telvariabele niet a priori als long integer gedeclareerd is. Basic is in het algemeen vrij soepel met de typetoekenning als de gebruiker dat niet heeft gedaan. De *default types* werken meestal zonder problemen, maar wees op uw hoede als de kans bestaat dat een positieve telgrootheid of integervariabele de 'magische' grens van 32767 ofwel 2^{15}-1 overschrijdt. Uiteraard kan men dergelijke problemen voorkomen door 'gevaarlijke' telgrootheden expliciet als het type *long* te declareren, bijvoorbeeld met een opdracht als DEFLNG I-K. Overigens blijft het oppassen, bijvoorbeeld bij het gebruik van de MOD-functie. Op deze functie wordt niet nader ingegaan, omdat de MOD-functie alleen in heel eenvoudige gevallen wordt gebruikt. Wie twijfelt aan de juistheid van een bepaalde Basic-bewerking kan het beste het een en ander proefondervindelijk uitzoeken aan de hand van een klein zelfgeschreven voorbeeld.

1.4 Coördinatenstelsels

In dit boek wordt uitgegaan van de VGA-modus met een resolutie van 640*480 pixels. De pixelposities zijn adresseerbaar als geheeltallige waarden van de coördinaten x en y, waarbij x van 0 tot 639 en y van 0 tot 479 loopt. Onthoud dat $x = 640$ en $y = 480$ voorbij de rand liggen en dus geen beeldpunten opleveren. Helaas voor wiskundigen loopt de y-as van boven naar beneden. Geen ramp overigens, mits men zich daarvan bewust is.

De WINDOW-opdracht maakt het mogelijk een eigen coördinatenstelsel in plaats van de beeldschermcoördinaten in te voeren. Wanneer de eenheden in horizontale en in verticale richting gelijk zijn, is het handig

de eigen (gebruikers-) coördinaten aan de afmetingen van het scherm aan te passen. Een goede keuze is:

```
WINDOW (-4,-3)-(4,3)
```

Hierbij ligt de oorsprong in het midden van het scherm en bovendien loopt de verticale as van beneden naar boven. Als de eigen coördinaten *u* en *v* worden genoemd, geldt het verband:

$u = 4\ (x/xm - 1) \qquad v = -3\ (y/ym - 1)$

waarbij $xm = 320$ en $ym = 240$.

Nadat de eigen coördinaten eenmaal met deze opdracht zijn ingevoerd, kunt u voor de verdere berekeningen de pixelcoördinaten *x* en *y* meestal wel vergeten.

Veelgebruikte grafische opdrachten zijn het plaatsen van een punt met PSET (X,Y) of het tekenen van een lijnstuk tussen twee gegeven punten LINE (X1,Y1)-(X2,Y2). De coördinaten worden opgegeven als de eigen coördinaten als deze daaraan voorafgaand gedefinieerd zijn, anders als pixelcoördinaten. Een aardige uitbreiding van de LINE-opdracht is:

```
LINE (X1,Y1)-(X2,Y2),,B
```

Hiermee wordt een rechthoek gevormd, waarbij (*x*1,*y*1) en (*x*2,*y*2) de overstaande hoekpunten zijn. Wanneer geldt dat $|x_1-x_2| = |y_1-y_2|$, heeft de lijn een helling van 45° en is de rechthoek een vierkant, tenminste als de bewerkingen in de VGA-modus plaatsvinden. Aan de LINE-opdracht kan ook nog een kleurwaarde toegevoegd worden en na toevoeging van de letter F ten slotte wordt de af te beelden rechthoek gevuld met kleur (BlockFill). Zo resulteert bijvoorbeeld de combinatie:

```
WINDOW (-4,-3)-(4,3)
LINE (-1,-1)-(1,1),4,BF
```

in een rood vierkant.

In bepaalde programma's kan het handig zijn een deel van het scherm te reserveren voor grafische bewerkingen en het resterende deel voor het (eventueel) weergeven van tekst. In de praktijk komt het vaak neer op het werken met een kleiner beeldscherm. Het gaat om de opdracht:

```
VIEW SCREEN (X1,Y1)-(X2,Y2),C
```

waarbij C een kleurwaarde is. Het kleinere (virtuele) grafische scherm is als het ware opgespannen tussen de punten (*x*1,*y*1) en (*x*2,*y*2), gedefinieerd in pixelcoördinaten. De toevoeging SCREEN betekent hier dat de grafische coördinaten, eventueel bepaald door een volgende WINDOW-opdracht, betrekking hebben op het volledige beeldscherm. Alle grafi-

sche uitvoer die buiten het virtuele scherm terecht zou komen, wordt niet afgebeeld, een techniek die als *clipping* bekend staat.

1.5 Tekst

Het kan gewenst zijn tijdens een grafische bewerking enige informatieve tekst op het scherm te plaatsen. Het hangt erg af van de persoonlijke voorkeur hoe en in welke mate men van die faciliteit gebruik wil maken. In onder andere QBB, paragraaf 6.4 en 10.2, vindt u desgewenst nadere informatie over dit onderwerp. Wie vooral in het maken van mooie plaatjes geïnteresseerd is, zal zuinig met tekst omspringen teneinde zoveel mogelijk ruimte te reserveren voor de grafische output. Het is daarom verstandig de tekst altijd zoveel mogelijk boven aan het scherm te plaatsen. Onder aan het scherm is ook mogelijk, maar helaas zijn de diverse Basic-dialecten daarover niet eensluidend. Helemaal links kan ook wel. Voor het plaatsen van tekst zijn er normaal gesproken altijd 25 regels van 80 kolommen beschikbaar. Voor wie nog Turbo Basic gebruikt is dat het maximum. In Power Basic, de opvolger van Turbo Basic, kan men in de VGA-modus tot 30 regels komen. Hetzelfde aantal kan ook in Q(uick)Basic bereikt worden. Vaak beginnen we een eerste mededeling op de eerste regel met

```
LOCATE 1,1 : PRINT"een stukje tekst"
```

Om dezelfde regel naderhand voor een andere mededeling te gebruiken kunt u de regel overschrijven, maar het is beter de regel helemaal te wissen met de opdracht:

```
LOCATE 1,1 : PRINT SPACE$(80) : LOCATE 1,1
```

Het getal 80 betekent natuurlijk het wissen van alle kolomposities, meer dan vaak nodig is, maar 80 is altijd goed. Om twee regels tegelijk te wissen gebruikt u SPACE$(160). Met de LOCATE-opdracht kan een grafische voorstelling van tekst worden voorzien. Probeert u eens het volgende mini-programma

```
SCREEN 9 : CLS
FOR I=1 TO 25 : LOCATE I,1 : PRINT I;
NEXT I
END
```

Wanneer alles goed gaat verschijnen links op het scherm onder elkaar de getallen 1 tot aan 25. Hetzelfde moet ook lukken in de text-modus. In de VGA-modus moet het schermgetal 12 zijn en kan het aantal regels tot 30 verhoogd worden. Wie echter nog het bijna verouderde Turbo Basic gebruikt zal helaas een foutmelding krijgen.

1.6 Structuur

In het algemeen worden in de programma's in dit boek voor lussen alleen de FOR ... NEXT-constructie en de DO ... LOOP-constructie gebruikt. Telgrootheden worden gewoonlijk met de letters I, J,..., N genoteerd. Bij een DO ... LOOP kunt u aan het begin of aan het eind nog een voorwaarde plaatsen, bijvoorbeeld:

```
DO WHILE INKEY$ = ""
    .................
LOOP
```

Dit wil zeggen dat de lus herhaald wordt totdat op een willekeurige toets wordt gedrukt. Om het programma echt op een willekeurig moment te kunnen stoppen, is het beter tussen de DO- en de LOOP-opdracht een extra opdracht op te nemen, bijvoorbeeld:

```
IF INKEY$ <> "" THEN END
```

of iets netter

```
IF INKEY$ <>"" THEN : SCREEN 0 : END
```

Een enkele maal kan het handig zijn een voorwaardelijke sprong van het type GOTO te gebruiken. Het is beter de GOTO-opdracht weinig te gebruiken en alleen toe te passen in die gevallen dat het de beknoptheid en de duidelijkheid van het programma ten goede komt.

In Basic wordt vaak gebruik gemaakt van subroutines. Bij kleine en overzichtelijke programma's is dat niet bezwaarlijk, maar zodra programma's langer en ingewikkelder worden, kunnen subroutines aanleiding geven tot het onbedoeld dubbel benoemen van programmavariabelen, een bekende bron van programmeerfouten. In Basic-programma's zijn alle variabelen globaal, tenzij men anders specificeert.

De klassieke conditionele constructie is de IF ... THEN (... ELSE)-constructie. Deze opdracht kan op een of op meer regels uitgevoerd worden. Een meerregelige conditie moet altijd worden afgesloten met een END IF-opdracht. Heel handig in de moderne Basic-versies is de SELECT CASE-opdracht, waarbij op een zeer overzichtelijke wijze een beslissing genomen wordt op grond van de waarde van een bepaalde programmavariabele. Een uitvoerige beschrijving vindt u in QBB, paragraaf 3.5.

1.7 EGA-kleuren

Het onderwerp 'kleuren' kent veel haken en ogen, waarover veel handleidingen geen duidelijkheid verschaffen. Het gebruik van kleuren hangt

af van de modus en van het Basic-dialect. We beperken ons hier tot EGA (SCREEN 9) en VGA (SCREEN 12) met in beide gevallen de mogelijkheid van het gelijktijdige gebruik van 16 kleuren.

U moet zich voorstellen dat van elke pixel op het beeldscherm in het videogeheugen van de computer een zogenaamd kleurattribuut aanwezig is van vier bits, een getal dus van 0 (0000 binair) tot en met 15 (1111 binair). Dat getal, soms ook kleurwaarde genoemd, zorgt dat de elektronenstralen die de desbetreffende pixel doen oplichten, zodanig gecombineerd worden dat een bepaalde kleur ontstaat. Welke kleur dat wordt, hangt af van een palet dat of standaard (default) is of door de programmeur nader gedefinieerd kan worden. Voorlopig wordt het standaard palet aangehouden waarvan de attribuutwaarden in de linkerkolom van tabel 1.1 gegeven zijn. De andere kolommen zijn van belang bij de samenstelling van kleuren volgens de hierna te bespreken RGB (rood, groen en blauw) methode.

		EGA	**R**	**G**	**B**
0	zwart	0	0	0	0
1	blauw	1	0	0	42
2	groen	2	0	42	0
3	cyaan	3	0	42	42
4	rood	4	42	0	0
5	magenta	5	42	0	42
6	bruin	20	42	21	0
7	wit	7	42	42	42
8	grijs	56	21	21	21
9	lichtblauw	57	21	21	63
10	lichtgroen	58	21	63	21
11	lichtcyaan	59	21	63	63
12	lichtrood	60	63	21	21
13	lichtmagenta	61	63	21	63
14	geel	62	63	63	21
15	helwit	63	63	63	63

Tabel 1.1

Normaliter wordt met de opdracht PSET (X,Y),14 dus een geel punt afgebeeld. Bij veel programma's die kleurige plaatjes produceren, de Mandelbrot set bijvoorbeeld, berust de berekening in feite op de bepaling van de attribuutwaarde. Elke in aanmerking komende punt van het plaatje wordt onderworpen aan dezelfde soort berekening die uitein-

delijk tot een bepaald positief geheel getal leidt. Bij elke pixel (x,y) van het plaatje behoort dus een getal, zeg maar getal k. Dat getal kan worden gebruikt als het kleurattribuut, eventueel door het modulo 16 te reduceren tot een getal tussen 0 en 15. Bent u om esthetische redenen niet tevreden over de gebruikelijke volgorde van de standaardkleuren, dan kunt u op de volgende wijze te werk gaan. Stel dat u zes van de standaardkleuren (inclusief zwart) wilt gebruiken. De gewenste standaardkleuren legt u vast in een array COL%(I) waarbij de index i van 0 tot en met 5 loopt. De waarden van de array, bijvoorbeeld 0, 1, 9, 2, 10, 4, leest u in met behulp van een DATA-lijst en een READ-opdracht. Daarbij reserveert u de index 0 voor het attribuut 0 (zwart). De uiteindelijke opdracht luidt dan:

```
PSET (X,Y),COL%(K MOD 6)
```

Zolang u deze 16 standaardkleuren gebruikt, gaat alles goed ongeacht het Basic-dialect. Wenst u meer kleuren, dan moet u bij de VGA-modus letten op de verschillen in kleurbehandeling. In bijvoorbeeld Power Basic (of Turbo Basic) kunt u met behulp van de PALETTE-opdracht over 64 kleuren beschikken. Als bijvoorbeeld het attribuut 4 in een gele kleur moet resulteren, gebruikt u de opdracht:

```
PALETTE 4,62
```

De 64 zogenaamde EGA-kleuren omvatten ook de 16 standaardkleuren. In tabel 1.1 is aangegeven welke rangnummers de 16 standaardkleuren in de EGA-lijst hebben. De kleuren kunnen zichtbaar gemaakt worden met behulp van de programma's EGACOL1 en EGACOL2. Beide programma's werken in de EGA-modus (SCREEN 9) en zijn in diverse Basic-dialecten bruikbaar. In Power Basic of Turbo Basic kan men ook de VGA-modus met SCREEN 12 gebruiken, maar bij Q(uick)Basic lukt dat niet, daar is men aangewezen op de RGB-methode.

In het programma EGACOL1 wordt een gekleurde rechthoek getoond in een opeenvolging van de 64 EGA-kleuren. Bovendien wordt het rangnummer van de kleur op het scherm afgedrukt. Na een wachttijd van een seconde verschijnt de volgende kleur. Uiteraard kan men de wachttijd in het programma eenvoudig veranderen. De pauze-constructie maakt gebruik van de timer van de computer waarbij TIMER (enkele-precisie-variabele) het aantal seconden dat sinds middernacht is verlopen, retourneert.

```
REM ***show van de 64 EGA kleuren***
REM ***naam:EGACOL1***
   SCREEN 9 : CLS
   WINDOW (-2,-1.5)-(2,1.5)
   LINE (-1,-1)-(1,1),1,B
```

```
    PAINT (0,0),1 : I=1
    DO WHILE I<64 AND INKEY$=""
        LOCATE 1,1 : PRINT I
        PALETTE 1,I
        T=TIMER
        DO WHILE TIMER-T<1 : LOOP
        I=I+1
    LOOP
END
```

In het tweede programma ziet u telkens een opeenvolging van 16 EGA-kleuren, inclusief hun rangnummers:

```
REM ***de 64 EGA kleuren in groepen van 16***
REM ***naam:EGACOL2***
    SCREEN 9 : CLS
    K=0
    FOR I=1 TO 7
        X=32+I*72 : Y=140
        LINE (X-16,Y-16)-(X+16,Y+16),I,B
        PAINT (X,Y),I
    NEXT I
    FOR I=1 TO 7 : X=32+I*72 : Y=260
        LINE (X-16,Y-16)-(X+16,Y+16),7+I,B
        PAINT (X,Y),7+I
    NEXT I
DO WHILE K<64
    FOR I=1 TO 7
        LOCATE 6,9*I+3 : PRINT (I+K) MOD 64
        PALETTE I,(I+K) MOD 64
    NEXT I
    FOR I=1 TO 7
        LOCATE 15,9*I+3 : PRINT (I+K+7) MOD 64
        PALETTE I+7,(I+7+K) MOD 64
    NEXT I
    PALETTE 15,63
    T=TIMER
    DO WHILE TIMER-T<.5 : LOOP
    IF INKEY$<>"" THEN A$=INPUT$(1) : END
    K=K+1
    LOOP
END
```

1.8 RGB-kleuren

In Q(uick)Basic en in Power Basic 3.0 worden de mogelijkheden van de VGA-modus beter benut door een kleur additief samen te stellen uit de drie basiskleuren Rood, Groen en Blauw. De hoeveelheid van een sa-

menstellende kleur wordt bepaald door een getal tussen 0 en 63. In totaal zijn er dus 64*64*64 samengestelde kleuren mogelijk. Daartoe behoren onder andere de 16 standaardkleuren; in tabel 1.1 ziet u hoe die kleuren opgebouwd worden.

Wie voldoende heeft aan de 16 standaardkleuren, behoeft verder niets meer te doen en dient een PALETTE-opdracht te vermijden. Om minstens één andere kleur, uit een eigen palet, te gebruiken moet u een opdracht als:

```
PALETTE A,C&
```

uitvoeren. Dat wil zeggen dat het kleurattribuut A voortaan de nieuw gekozen kleur weergeeft waarbij C& de kleurwaarde is van de nieuwe kleur. De kleurcode wordt als volgt bepaald:

```
C& = R + 256*G + 65536*B
```

waarbij r, g en b de hoeveelheden zijn van de rode, groene en blauwe component. Omdat een integer beperkt is tot waarden < 32768 heeft de kleurwaarde het type van een long integer.

Het definiëren van een aantrekkelijk kleurenpalet is niet gemakkelijk, er zijn te veel mogelijkheden. Het programma MIXER biedt de mogelijkheid om op interactieve wijze een kleur samen te stellen uit verschillende componenten. Er verschijnt een gekleurde cirkel met op de bovenste regel de hoeveelheden van de R-, G- en B-component. Met de functietoetsen F1 en CTRL-F1 vergroot of verkleint u de intensiteit van de R-component. Hetzelfde principe geldt voor de G-component met de functietoetsen (CTRL) F2 en voor de B-component met de functietoetsen (CTRL) F3. Het programma kan worden gestopt met een van de toetsen F7 tot en met F10.

```
REM ***mengen van kleuren in Q(uick)Basic***
REM ***naam:MIXER***
    SCREEN 12: CLS
    XM = 320: YM = 240
    R = 32: G = 32: B = 32
    LOCATE 1, 1
    PRINT "toets F1,F2 en F3 voor verhoging R,G en B component"
    LOCATE 3, 1
    PRINT "gebruik zelfde toetsen met CTRL voor verlaging"
    LOCATE 5, 1: PRINT "toets F7 enz. voor einde programma"
    LOCATE 7, 1
    PRINT "druk op een willekeurige toets voor vervolg"
    A$ = INPUT$(1): CLS
    DO
        COL& = R + 256 * G + 65536 * B
        PALETTE 1, COL&
```

```
        CIRCLE (XM, YM), 160, 1: PAINT (XM, YM), 1
        LOCATE 2, 18: PRINT R
        LOCATE 2, 38: PRINT G
        LOCATE 2, 58: PRINT B
        A$ = "": A$ = INKEY$
        IF LEN(A$) > 1 THEN
     CS = ASC(MID$(A$, 2, 1))
     SELECT CASE CS
     CASE 59
        R = R + 1: R = R MOD 64
     CASE 60
        G = G + 1: G = G MOD 64
     CASE 61
        B = B + 1: B = B MOD 64
     CASE 94
        R = R - 1
        IF R < 0 THEN R = R + 64
     CASE 95
        G = G - 1
        IF G < 0 THEN G = G + 64
     CASE 96
        B = B - 1
        IF B < 0 THEN B = B + 64
     CASE 65, 66, 67, 68
        END
     END SELECT
        END IF
    LOOP
END
```

Soms loont het de moeite een kleurenpalet samen te stellen als een soort kleurenwaaier tussen twee kleuren in. Als u uitgaat van een 'beginkleur' C&(1) en een 'eindkleur' C&(2) en u wilt in totaal m kleuren gebruiken, kunt u de tussenliggende kleuren definiëren met de lineaire interpolatieformule:

```
C&(J) = C&(1) + (C&(2) - C&(1))*(J - 1)/(M - 1)
```

waarbij J van 1 tot en met m loopt (uiteraard met $m \leq 16$).

1.9 Varia

De programma's in het boek kunnen soms opdrachten bevatten die niet essentieel voor de werking van het programma zijn, maar die wel het gebruikersgemak kunnen bevorderen. Dat kan bijvoorbeeld een pauze of voorwaardelijke stop zijn waarbij het programma even tot stilstand komt. Er zijn daartoe de volgende twee mogelijkheden.

De opdracht

```
A$ = INPUT$(1)
```

betekent dat het programma wacht op het inlezen van een symbool van het toetsenbord. Pas nadat een willekeurige toets (bijvoorbeeld de spatiebalk) ingedrukt is, vervolgt het programma zijn weg. Het door de toets gegenereerde symbool, de scancode, wordt opgenomen als de stringvariabele A$ maar vaak wordt daar niets mee gedaan. Een andere mogelijkheid is

```
DO : LOOP UNTIL INKEY$ <>""
```

Sommige programma's kunnen veel tijd in beslag nemen, vooral bij oudere computers met nog een 8086 of 80286 processor. Veel is daar niet aan te doen, maar soms helpt het een paar parameters in het programma gunstiger waarden te geven zodat minder pixels gebruikt behoeven te worden. Vooral QBasic kan erg traag zijn. Blijkbaar hoopt de firma Microsoft dat de gebruiker daarom de snellere compiler-versie QuickBasic zal gaan aanschaffen. Overigens is de nieuwste Power Basic versie 3.0 weer sneller en nog beter.

Wanneer men niet alleen geïnteresseerd is in het eindresultaat van een programma maar wanneer men ook de ontstaanswijze van een grafische figuur op het scherm wil volgen, kan een programma te snel zijn. Men kan dan in het programma een vertraging inbouwen, een lusconstructie bijvoorbeeld waarbij de computer even bezig gehouden wordt met iets onnozels als het printen van een lege string, bijvoorbeeld

```
FOR I=0 TO 100
   LOCATE 1,1
   PRINT ""
NEXT I
```

Een ervaren programmeur kan er een modulaire pauze-procedure van maken, waarbij het aantal malen dat de lus doorlopen wordt als parameter aangeroepen kan worden. De grootte van de lus moet proefondervindelijk vastgesteld worden.

Er bestaan wel opdrachten als DELAY en SLEEP waarbij een vertragingstijd in seconden of milliseconden gegeven moet worden, maar omdat die opdrachten in de verschillende Basic versies niet dezelfde betekenis hebben zijn ze in dit boek vermeden. Ook kan men gebruik maken van een lusconstructie als

```
A = TIMER
DO : LOOP WHILE TIMER-A<1.5
```

TIMER is een getal met enkele precisie en retourneert het aantal

seconden dat sinds middernacht is verstreken. A geeft het begin van de pauze aan, na bijvoorbeeld anderhalve seconde hervat het programma kennelijk zijn gewone gang.

De meeste programma's eindigen gewoon met END. In Q(uick)Basic verschijnt dan de mededeling "`Press any key to continue`". Wie dit storend vindt kan als voorlaatste opdracht een pauze-opdracht DO : LOOP UNTIL INKEY$<>"" inlassen. Datzelfde kan ook nuttig zijn wanneer men van een programma een EXE-file wil maken. Het kan namelijk gebeuren dat het plaatje na beëindiging van het programma meteen verdwijnt. Bij grafische programma's waarbij veel vaak onzichtbaar werk verzet moet worden, kan ook een BEEP-opdracht goede diensten bewijzen als akoestisch signaal dat het programma klaar is of althans in een afsluitende fase verkeert. Sommige handleidingen adviseren om aan het einde van een grafisch programma terug te keren naar het tekstscherm, dus met de opdracht SCREEN 0 (zo nodig aangevuld met WIDTH 80). Het wordt aan de gebruiker overgelaten deze opdrachten al of niet in de hier geboden programma's op te nemen.

1.10 Wiskundige conventies

Vooral in grafische programma's wordt veel gebruik gemaakt van wiskundige functies en formules. In een wiskundige formulering gebruikt men veel speciale notaties waarbij allerlei vreemde symbolen voorkomen, de *grieken* volgens de typograaf. Ook is een wiskundige tekst ruim voorzien van indices onder en boven, de zogenaamde subscripts en superscripts. Daarentegen wordt in een opdrachtregel van een programma alles op dezelfde regel geplaatst, en verder wordt in een programmeertaal meer gebruik gemaakt van haakjes. In de hier gegeven programma's maken we om typografische redenen veel gebruik van hoofdletters, maar in de begeleidende tekst vervangen we die vaak door kleine letters, eventueel voorzien van subscript of superscript. Wordt in de wiskundige tekst bijvoorbeeld x_1 gebruikt, dan ziet u dat in het programma terug als X1. Bij de weergave van programma's volgen we in het boek in hoofdzaak de conventies van Power Basic. De gebruikelijke lay-out van (Quick)Basic met de vele spaties is minder geschikt voor programma's met veel wiskundige berekeningen. Omdat in dit boek nergens de logaritmen met grondtal 10 worden gebruikt, betekent log evenals in een programmeertaal gewoon de natuurlijke logaritme waarvan het grondtal $e = 2.7182818...$ is. In plaats van e^x schrijft men meestal $\exp(x)$, een notatie die overeenstemt met de notatie in Basic. In Basic is $\pi = 3.141593...$ geen bekende, tot de taal behorende, constante. Hebt u in

een programma π nodig, dan dient de constante aan het begin van het programma apart ingevoerd te worden. Een goede methode is:

```
PI = 4*ATN(1)
```

Deze formule berust erop dat de tangens van 45° (π/4 in radialen) precies 1 is. Deze notatie is onafhankelijk van het type. Is PI een getal met dubbele precisie, dan is deze notatie korter dan bijvoorbeeld PI = 3.14159265358979..., een uitdrukking die niet iedereen paraat heeft.

2 Eenvoudige programma's

2.1 ASCII-symbolen

Dit hoofdstuk begint met een aantal eenvoudige programma's die te maken hebben met de ASCII-symbolen. Deze symbolen zijn afkomstig van de Amerikaanse telecommunicatie, en zijn uitgebreid met enkele andere symbolen tot een reeks van 256. Daaronder bevinden zich de gewone letters van het alfabet, de cijfers en een aantal veelgebruikte speciale symbolen. Er zijn ook wonderlijke symbolen bij, zoals een lachend gezichtje. Daarentegen ontbreken symbolen die men juist zou verwachten. De getallen tussen 0 en 255 corresponderen precies met acht bits of een byte. Niet voor alle waarden wordt een symbool afgedrukt, sommige waarden laten de apparatuur een actie uitvoeren zoals het rinkelen van een bel of 'ga naar het begin van de volgende regel' voor een printer. Hoe het ook zij, die ASCII-reeks is de standaard symbolenverzameling waar de rekenmachines op ingesteld zijn. Om een willekeurig ASCII-symbool op het scherm te zetten, gebruikt u de opdracht:

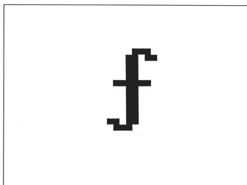
```
PRINT CHR$(N)
```

waarbij *n* het rangnummer is. Zo correspondeert *n* = 227 met de Griekse letter π. Wanneer voor de gegeven waarde van *n* geen symbool bestaat, wordt er niets afgedrukt.

Het eerste programma ASCPRINT maakt van het door *n* bepaalde symbool een vergrote versie. De vergrotingsfactor kunt u zelf kiezen als een getal tussen 2 en 20. Het aanvankelijk eventueel afgedrukte ASCII-symbool is een bitpatroon in een staand rechthoekig veld van 8 bij 16 pixels in de VGA-modus. Het VGA-schermbeeld van 640:480 pixels is dienovereenkomstig opgebouwd uit 30 rijen van 80 kolommen. Het bitpatroon van een enkel teken wordt afgetast met de opdracht POINT (X,Y)

die signaleert of een pixel van het rechthoekige raster tot het symbool behoort of niet. Vervolgens wordt afhankelijk van de factor de pixel als een blokje in het midden van het scherm weergegeven.

```
REM ***het schrijven van een groot ASCII symbool***
REM ***naam:ASCPRINT***
   SCREEN 12 : CLS
   INPUT"ASCII getal = ",N : CLS
   PRINT"geef vergroting als getal tussen 2 en 20"
   INPUT"vergroting = ",F : CLS
   LOCATE 1,1 : PRINT CHR$(N)
   XC=320-5*F : YC=200-8*F
   FOR I=0 TO 7 : FOR J=0 TO 15
      A=POINT (I,J)
      X1=XC+F*I : Y1=YC+F*J
      X2=X1+F-1 : Y2=Y1+F-1
      IF A<>0 THEN LINE (X1,Y1)-(X2,Y2),,BF
   NEXT J : NEXT I
   LINE (XC,YC)-(XC+8*F,YC+16*F),12,B
END
```



Figuur 2.1 ASCII-waarde 159, het guldenteken

Het tweede programma werkt alle 255 mogelijkheden af. De vergrotingsfactor is vast op 20 gezet, zodat het symbool maximaal een rechthoek van 160*320 pixels beslaat. In het programma is een pauze van een halve seconde opgenomen.

```
REM ***reeks van vergrote ASCII symbolen***
REM ***naam:ASCRIJ***
   SCREEN 12 : CLS
   XC=220 : YC=60
```

```
    FOR N=1 TO 255
        LOCATE 1,1 : PRINT CHR$(N)
        LOCATE 1,20 : PRINT N
        FOR I=0 TO 7 : FOR J=0 TO 15
            LINE (220,60)-(380,380),4,B
            A=POINT (I,J)
            X1=XC+20*I : Y1=YC+20*J
            X2=X1+19 : Y2=Y1+19
            IF A<>0 THEN LINE (X1,Y1)-(X2,Y2),,BF
            IF INKEY$<>"" THEN END
        NEXT J : NEXT I
        T=TIMER
        DO WHILE TIMER-T<.5
        LOOP : CLS
    NEXT N
END
```

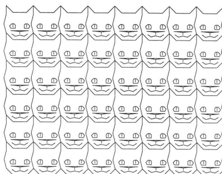
Het derde en laatste programma WOORD van deze serie maakt op analoge wijze een vergrote versie van een gegeven woord. De horizontale en verticale vergrotingsfactoren, in het programma hor en ver genoemd, kunnen daarbij verschillend zijn. Gekozen is het woord QBasic, maar het kan in principe elk ander woord zijn.

```
REM ***van een woord kan een vergrote kopie gemaakt worden***
REM ***vorm van letters is bepaald door HOR en VER***
REM ***positie van woord is bepaald door XSTART en YSTART***
REM***naam:WOORD***
    W$="QBASIC"    'gegeven woord
    HOR=4 : VER=6
    XSTART=64 : YSTART=64
    SCREEN 1 : CLS
    LOCATE 1,1 : PRINT W$
    FOR X=0 TO 8*LEN(W$)
        FOR Y=0 TO 15
            COL=POINT (X,Y)
            X1=XSTART+HOR*X : Y1=YSTART+VER*Y
            LINE (X1,Y1)-(X1+HOR-1,Y1+VER-1),COL,BF
        NEXT Y
    NEXT X
END
```

2.2 Een tegelwand van katten

Bepaalde grafische programma's vereisen veel voorbereidend werk. Om een bepaald plaatje, de kop van een kat bijvoorbeeld, op het beeldscherm te zetten zou men kunnen beginnen met een basistekening op ruitjespapier. Hierbij zijn de samenstellende lijnen gebroken rechte lij-

nen met hoekpunten op de roosterpunten. Elke ononderbroken lijn $P_1P_2P_3$, ... vertegenwoordigt dan een rij *x*-coördinaten en *y*-cöordinaten die als gehele getallen uitgedrukt kunnen worden. Op een soortgelijke wijze wordt in het programma KATTEN de kop van een kat opgebouwd uit zeven gebroken lijnen. De array *m*(.) geeft aan uit hoeveel lijnstukjes elke lijn bestaat. De coördinaten van de hoekpunten zijn verenigd in de dubbele arrays *x*(.,.) en *y*(.,.). Ten slotte wordt de kattekop op verschillende schermposities afgebeeld op de wijze van een tegeltableau dat aan het werk van de bekende graficus Maurits Escher doet denken.



Figuur 2.2 De kop van een kat als tegelversiering

```
REM ***een tegelwand van katten op de wijze van Escher***
REM ***naam:KATTEN***
   SCREEN 12 : CLS
   WINDOW (6,6)-(114,87)
   DIM X(7,13),Y(7,13),M(7)
REM ***relatieve coordinaten kattenmotief***
   M(1)=13 : M(2)=7 : M(3)=7 : M(4)=6
   M(5)=2 : M(6)=2 : M(7)=5
   DATA 0,0,-1,4,0,8,0,12,3,9,9,9,12,12,12,8,11,4,12
   DATA 0,9,-3,3,-3,0,0
   DATA 2,2,1.5,3,2.5,4,3.5,4,4,3.5,4,2,2,2
   DATA 8,2,8,3.5,8.5,4,9.5,4,10.5,3,10,2,8,2
   DATA 2,1,10,1,6,.6,6,-.5,6,.6,2,1,3,4,3,2.5,9,4,9,2.5
   DATA 2.5,-.5,4,-1.5,6,-1,8,-1.5,9.5,-.5
   FOR I=1 TO 7 : FOR J=1 TO M(I)
      READ X(I,J),Y(I,J)
   NEXT J : NEXT I
   FOR I=1 TO 6
      FOR J=1 TO 8
```

```
        FOR K=1 TO 7
          PSET (12*J+X(K,1),12*I+Y(K,1))
          FOR L=2 TO M(K)
            LINE -(12*J+X(K,L),12*I+Y(K,L))
          NEXT L
        NEXT K
      NEXT J
    NEXT I
END
```

2.3 Draaiende cirkels

Nu volgt een aantal eenvoudige grafische programma's met behulp waarvan bepaalde krommen, soms heel fraaie, op het beeldscherm gebracht kunnen worden. In de regel kiezen we het benodigde coördinatenstelsel zo eenvoudig mogelijk, liefst met de oorsprong in het midden van het beeld.

De eenvoudigste figuur is een cirkel met straal r. Is $O(0,0)$ de oorsprong en $P(x,y)$ een willekeurig punt van de cirkel, dan geldt dus $OP = r$. Volgens de afstandformule is $x^2 + y^2 = r^2$. Deze formule is echter niet direct bruikbaar in een programma. Het is beter een andere, equivalente wiskundige formulering te gebruiken, een zogenaamde parametervoorstelling:

(2.1) $x = r \cos(t)$ $y = r \sin(t)$

waarbij t een reëel getal is. Voor elke waarde van t gelden de waarden (x,y), die meetkundig met een punt P op de cirkel corresponderen. Loopt t van 0 tot aan 2π, dan doorloopt P precies de hele cirkelomtrek. We geven daarbij aan dat we t in radialen rekenen, de natuurlijke eenhedenkeuze van de sinus- en cosinusfunctie. Soms is het prettig om in meer vertrouwde graden te rekenen en dan is de omrekening eenvoudig

180 graden $\approx \pi$ radialen

De formule (2.1) wordt in heel veel programma's toegepast, soms in de uitgebreide vorm:

$$x = x_0 + r \cos(t) \qquad y = y_0 + r \sin(t)$$

Met deze formule kan een cirkel met straal r getekend worden, waarvan (x_0,y_0) het middelpunt is.

Er zijn in principe twee manieren om een kromme lijn op het scherm af te beelden: als een stippellijn, een rij van losse puntjes, of als een aan-

eenschakeling van kleine lijnstukjes. De eerste methode wordt beschreven in de volgende programmaregels:

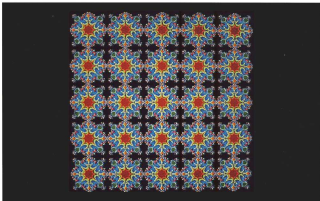
```
..........................
FOR I=0 TO N-1
   T=2*I*PI/N
   X=R*COS(T) : Y=R*SIN(T)
   PSET (X,Y)
NEXT I
..........................
```

De tweede methode wordt gevolgd in deze regels:

```
..........................
FOR I=0 TO N
   T=2*I*PI/N
   X=R*COS(T) : Y=R*SIN(T)
   IF I=0 THEN PSET (X,Y) ELSE LINE -(X,Y)
NEXT I
..........................
```

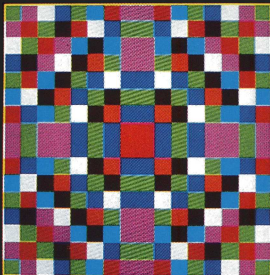
Het programma CIRKELB vormt een kronkellijn als een samengestelde beweging van twee cirkels. U kunt zich de eerste cirkel voorstellen als een draaischijf. Een aan de draaischijf verbonden punt is zelf weer het middelpunt van een tweede draaiende cirkel. De draaischijven draaien met verschillende snelheden in de verhouding *a*. Ook de stralen van de twee schijven kunnen verschillen met een factor *r*. Het programma gedraagt zich als een zich eindeloos herhalende lus, een DO ... LOOP, die beëindigd wordt zodra u op een willekeurige toets drukt. De parameter *t* neemt daarbij telkens met een kleine waarde *h* toe. In het programma zijn voor *a*, *r* en *h* geschikte waarden genomen, maar het loont de moeite te experimenteren met andere waarden. Kies bij voorkeur voor *a* een breuk als een verhouding van twee niet te grote gehele getallen.

```
REM ***samengestelde cirkelbeweging***
REM ***naam:CIRKELB***
   SCREEN 12 : CLS
   WINDOW (-3.2,-2.4)-(3.2,2.4)
   A=11/7 : R=.6 : H=.02
   LOCATE 1,1 : PRINT"druk op een toets voor einde programma"
   DO
      X=COS(T) : Y=SIN(T)
      X1=X+R*COS(A*T) : Y1=Y+R*SIN(A*T)
      PSET (X1,Y1),14
      T=T+H
REM ***vertragingslus***
     FOR I=1 TO 100 : LOCATE 1,1 : PRINT "" : NEXT I
  LOOP UNTIL INKEY$<>""
END
```

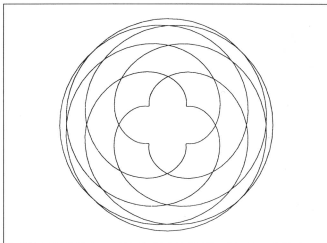












Figuur 2.3 Een samengestelde cirkelbeweging

2.4 Fourier en Lissajous

Volgens de wetten van de natuurkunde kunnen periodieke verschijnselen beschouwd worden als een combinatie van zogenaamde harmonische golven of trillingen. De voortplanting van licht en geluid, radio en televisie berust hierop. Het eenvoudigste periodieke verschijnsel, een harmonische trilling, wordt beschreven door een sinus-functie $y = a \sin(ct+\alpha)$ (met $a > 0$), waarbij t de betekenis van tijd heeft, in seconden bijvoorbeeld. De sinus-functie $\sin(x)$ is periodiek met de periode 2π en dus heeft y de periode $T = 2\pi/c$. Het getal c wordt de cirkelfrequentie genoemd. Doorloopt t het interval $[0,T]$, dan voert y een complete trilling uit, te beginnen bij $a \sin(\alpha)$. De grootheid α is de fasehoek en heeft meestal een waarde tussen 0 en 2π. Soms wordt in plaats van α het fasegetal $\alpha/(2\pi)$ gebruikt, een getal tussen 0 en 1. De sinus-functie $\sin(x)$ neemt waarden aan tussen -1 en 1. De functie $y = a \sin(ct+\alpha)$ neemt dus waarden aan tussen $-a$ en a. Het getal a wordt de amplitude genoemd. De eenvoudigste periodieke functies met de periode 2π zijn 1, $\sin(x)$, $\cos(x)$, $\sin(2x)$, $\cos(2x)$, $\sin(3x)$, enzovoort, en algemeen $\sin(mx)$, $\cos(mx)$ waarbij m een natuurlijk getal is. Volgens een beroemde wiskundige stelling genoemd naar Fourier, een Franse onderzoeker uit de vorige eeuw die zich in het bijzonder met de theorie van de warmtegelei-

ding bezighield, kan elke periodieke functie $f(x)$ opgevat worden als een oneindige reeks van dergelijke elementaire harmonische functies.

$$f(x) = a_0 + a_1\cos(x) + b_1\sin(x) + a_2\cos(2x) + b_2\sin(2x) + \ldots$$

Omdat we niet met oneindig veel termen kunnen rekenen, moet de Fourier-reeks ergens afgebroken worden, bijvoorbeeld na honderd of duizend termen. De afgebroken reeks geeft dan een benadering van de functie $f(x)$ als de som van een reeks met oneindig veel termen.

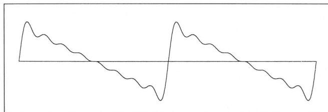
```
REM ***grafiek van een trigonometrische reeks***
REM ***naam:FOURIER***
    SCREEN 12 : CLS : PI=4*ATN(1)
    WINDOW (-8,-3)-(8,3)
    M=8 : DIM X(M)
    FOR I=1 TO M
        B(I)=1/I
    NEXT I
    JMAX=1000   'aantal punten
    LINE (-2*PI,0)-(2*PI,0)
    FOR J=0 TO JMAX
        X=-2*PI+4*PI*J/JMAX
        S=0
        FOR K=1 TO M
            S=S+B(K)*SIN(K*X)
        NEXT K
        IF J=0 THEN PSET (X,S) ELSE LINE -(X,S)
    NEXT J
END
```

Het programma FOURIER laat zien wat het effect is van het afbreken van een bepaalde Fourier-reeks, waarbij $f(x)$ een soort zaagtandfunctie is. Op de wiskundige aspecten, hoe interessant ook, wordt hier niet ingegaan. Maar uit een herhaling van hetzelfde programma met afbreken op verschillende plaatsen blijkt op experimentele wijze gauw genoeg wat er aan de hand is. Het programma demonstreert tevens hoe u met behulp van de computer een functie die door een reeks van termen gedefinieerd is, als een grafiek voorstelt. De reeks van het voorbeeld is:

$$f(x) = \sin(x)/1 + \sin(2x)/2 + \sin(3x)/3 + \sin(4x)/4 + \ldots$$

In beginsel loopt deze reeks oneindig door, maar het programma stopt na m termen waarbij m (als integer) willekeurig ingesteld kan worden. Het getal π is aan het begin van het programma gedefinieerd als PI = 4*ATN(1), voor sommigen een toverformule, voor anderen met wat meer wiskundige kennis de uitdrukking van de eigenschap dat $\tan(\pi/4) = 1$ is. De kern van het programma is een dubbele lus. In de buitenste lus worden 1000 x-waarden in het interval $[-2\pi,2\pi]$ geselecteerd, in de binnenste lus wordt voor elke x de som van m termen van de reeks gevormd.

In figuur 2.4 is het grafische resultaat weergegeven voor $m = 8$. Herhaalt u de berekening voor bijvoorbeeld $m = 80$, dan lijkt het resultaat duidelijker op een zaagtandfunctie. Het valt op dat de benadering van de zaagtandfunctie slecht is bij de sprongen op de plaatsen $x = 0$, $x = \pm 2\pi$. Dat is overigens een bekend effect, het zogenaamde verschijnsel van Gibbs in de schakeltechniek.



Figuur 2.4 De benadering van een zaagtand

De natuurkundige Lissajous deed experimenten met een aan een lang koord opgehangen schrijvende pen, een slinger die zich zowel in de x-richting als in een loodrecht daarop staande y-richting kon bewegen. Een (harmonische) slingering in alleen de x-richting wordt in dit geval beschreven door:

$$x = a \sin(ct + \alpha)$$

Hieraan kunnen we overeenkomstige waarden voor de y-richting toevoegen:

$$y = b \sin(ct + ß)$$

Bedenk wel dat voor de frequenties in beide richtingen gelijke waarden zijn gekozen, maar dat de amplitude en de fase kunnen verschillen. De samengestelde beweging, hetzij in het fysische experiment, hetzij in een computermodel, blijkt in het algemeen een ellips op te leveren. In het computerexperiment kunnen we de frequenties in de x-richting en de y-richting verschillend nemen en dan kunnen er interessante en soms fraaie kromme lijnen ontstaan. Door de eenheden van tijd en lengte een beetje handig te kiezen kunnen we het wiskundige model wat vereenvoudigen. Zonder in te gaan op de details kunnen we uitgaan van:

$$x = a \sin(t) \qquad y = \sin(st + \varphi)$$

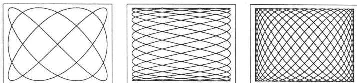
zonder verlies van algemeenheid. Omdat het scherm meestal rechthoekig is in de verhouding 4:3, stellen we $a = 4/3$.

De frequentieverhouding *s* wordt opgevat als een breuk *n/m*, waarbij *m* en *n* eenvoudige natuurlijke getallen zijn.

Bij de experimenten komt het vooral aan op de keuze van *s*. Hoe groter *m* en *n* zijn, des te ingewikkelder is de uiteindelijke baan van de 'schrijfstift' *P*(*x*,*y*), maar dat is altijd een kromme die zich na een aantal lussen weer sluit. Minder belangrijk is de keuze van het faseverschil φ, en meestal kunt u volstaan met een fasegetal van 0, ±.25 en .5. Het programma kan desgewenst met een enkele toetsaanslag worden beëindigd.

```
REM ***Lissajous kromme***
REM ***naam:LISSA***
   SCREEN 12 : CLS : PI=4*ATN(1)
   WINDOW (-2,-1.5)-(2,1.5)
   A=4/3   'vormfactor
   M=15   'eerste frequentie
   N=22   'tweede frequentie
   F=.5   'faseverschil
   S=N/M : F=2*PI*F : H=.005 : T=0
   DO WHILE INKEY$="" AND T<2*M*PI
      X=A*SIN(T)
      Y=SIN(S*T+F)
      PSET (X,Y)
      T=T+H
   LOOP : BEEP
END
```

Een voorbeeld van dergelijke Lissajous-figuren is weergegeven in figuur 2.5.



Figuur 2.5 Figuren van Lissajous

Het volgende programma LISSAX is een uitbreiding van het voorgaande programma. De wiskundige kern is dezelfde, maar nu worden bij wijze van demonstratie achter elkaar drie verschillende Lissajous-krommen afgebeeld. Dat betekent in hoofdzaak wat meer administratie. De drie verschillende gevallen worden gekozen met behulp van de SELECT CASE-constructie, waarbij *j* de teller is van de hoofdlus. Nadat het

plaatje gemaakt is, wacht het programma een aantal seconden, in dit geval drie, voordat het aan het volgende begint. We herhalen nog even dat een pauze-opdracht ook op eenvoudiger wijze kan worden uitgevoerd, maar dat dit in de verschillende Basic-dialecten niet op dezelfde wijze mogelijk is. De hier gevolgde methode is ietwat omslachtig maar wel onafhankelijk van het gekozen dialect.

```
REM ***figuur van Lissajous***
REM ***naam:LISSAX***
    SCREEN 12 : CLS : PI=4*ATN(1)
    WINDOW (-2.4,-1.8)-(2.4,1.8)
    H=PI/400 : AMPL1=1.6 : AMPL2=1.2
    FOR J=1 TO 3
        SELECT CASE J
            CASE 1
                M=3 : N=4 : FASE=0
            CASE 2
                M=8 : M=13 : FASE=.5
            CASE 3
                M=34 : N=55 : FASE=.25
            END SELECT
        FREQ=M/N : FASE=FASE*2*PI
        LOCATE 2,15 : PRINT"M = ";M
        LOCATE 2,30 : PRINT"N = ";N
        LINE (-1.1*AMPL1,-1.1*AMPL2)-(1.1*AMPL1,1.1*AMPL2),,B
        PSET (0,AMPL2*SIN(FASE))
        FOR K=0 TO 800*N : T=K*H
            X=AMPL1*SIN(FREQ*T) : Y=AMPL2*SIN(T+FASE)
            LINE -(X,Y)
        NEXT K : GOSUB wacht
        IF J<3 THEN CLS
    NEXT J
    LOCATE 2,55 : PRINT"einde programma"
END
wacht:
    A=TIMER
    DO
    LOOP WHILE TIMER-A<3
RETURN
```

Lissajous-figuren kunnen op allerlei wijzen gevarieerd worden. De wiskundige formulering berust op het combineren van een paar sinus- en cosinus-functies. Ook in dit geval worden x en y voorgesteld door de functies $f(t)$ en $g(t)$ die opgebouwd zijn uit een of meer goniometrische functies

$$x = f(t) \qquad y = g(t)$$

Bij de Lissajous-krommen waren *f* en *g* een enkele sinus- of cosinusfunctie, maar er is niets op tegen voor *f* en *g* een aantal van die functies te gebruiken. In het programma LISSAV is slechts een van de vele mogelijkheden verwerkt:

$x = a \cos(t) + b \cos(st)$
$y = \sin(t)$

De constanten *a*, *b* en *s* zijn vrij willekeurig gekozen. Het is leuk hetzelfde programma uit te voeren met andere, zelf gekozen waarden. Ook kan men in plaats van een cosinus-functie een sinus-functie nemen of omgekeerd. Voor $g(t)$ is nu nog een enkele sinus gekozen, maar ook hiervoor kan men iets algemeners kiezen, bijvoorbeeld een som van twee termen als $f(t)$ maar dan met andere constanten, bijvoorbeeld:

$x = a \cos(t) + b \sin(st)$
$y = c \sin(t) + d \cos(st)$
enzovoort.

In het programma wordt gebruik gemaakt van een lus van onbepaalde lengte. Het programma stopt nadat een willekeurige toets is ingedrukt.

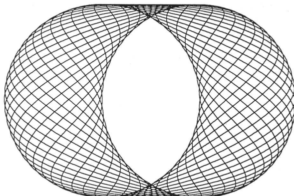
```
REM ***Lissajous-achtige krommen***
REM ***name:LISSAV***
    SCREEN 12 : CLS
    WINDOW (-2,-1.5)-(2,1.5)
    A=1 : B=.5 : S=2.45  'parameters
    T=0 : H=.01
    LINE (-A-B-.1,-1.1)-(A+B+.1,1.1),,B
    PRINT"druk op een willekeurige toets voor einde programma"
    DO WHILE INKEY$=""x  AND T<315
        X=A*COS(T)+B*COS(S*T)
        Y=SIN(T)
        IF T=0 THEN PSET (X,Y) ELSE LINE -(X,Y)
        T=T+H
    LOOP
    LOCATE 1,1 : PRINT SPACE$(80)
END
```

Het resultaat van het programma is afgebeeld in figuur 2.6.

2.5 Computeranimatie

Het laatste programma in dit hoofdstuk demonstreert het basisprincipe van computeranimatie. Het is de bedoeling een figuurtje, een sterretje bijvoorbeeld, over het beeldscherm te bewegen in een snelle opeenvolging van afzonderlijke plaatjes. De eerste stap is het definiëren en af-

beelden van het te bewegen figuurtje. In het modelprogramma ANIMA wordt dit gerealiseerd door in de linkerbovenhoek van het scherm een gekleurd cirkelschijfje te plaatsen. De volgende stap is het opbergen van de data van het plaatje in een array. Deze wordt FIGURE% genoemd met het symbool % voor integerwaarden. Van te voren moet ruimte worden gereserveerd met een DIM-opdracht en dit verdient extra aandacht.



Figuur 2.6 Een Lissajous-achtige kronkellijn

Denk u in dat het plaatje wordt geplaatst in een rechthoek van horizontaal *h* en verticaal *v* pixels. Omdat de computer de data verwerkt in de vorm van bytes, groepjes van acht bits, moet *h* worden omgerekend in bytes. De formule is 1 + int(*h*/8) zodat we in het gegeven geval met *h* = 41 en *v* = 41 op een blokje van 6*41 bytes uitkomen. Werkt u in de VGA-modus, en daar wordt ook van uitgegaan, dan moet het geheel nog met de factor 2 vermenigvuldigd worden. Ten slotte moeten er bij het totaal nog vier bytes opgeteld worden voor de array-administratie. In formulevorm luidt dit:

grootte in bytes = 4 + 2 * aantal bytes in blokje

In het gegeven programma komen we aldus uit op een totaal van 496 te reserveren array-plaatsen.

Wie niet zo handig is in het hanteren van dergelijke formules, doet er verstandig aan royaal te zijn bij het toekennen van ruimte aan de array. Meestal is er voldoende geheugenruimte en kunt u beter wat meer reserveren dan echt nodig is. Als er te weinig ruimte gekozen is, verschijnt

vaak een foutmelding of loopt het programma vast met een zwart scherm waarop niets meer gebeurt. Het motiefje wordt opgeborgen met de opdracht:

```
GET (X1,Y1)-(X2,Y2),FIGURE%
```

Hierbij worden de coördinaten van de hoekpunten linksboven en rechtsonder van de rechthoek opgegeven. Zodra het te bewegen figuurtje in een array opgenomen is, kan het op een willekeurige positie op het beeldscherm neergezet worden. De positie (*x*,*y*) wordt daarbij altijd bepaald door de linkerbovenhoek van het rechthoekige blokje waarin het sterretje geplaatst is. De afbeelding wordt naar het scherm gekopieerd met de opdracht:

```
PUT (X,Y),FIGURE%
```

Hierbij is nog iets eigenaardigs aan de hand dat essentieel is bij computeranimatie. Op de plaats waar het cirkelschijfje verschijnt, wordt de reeds aanwezige informatie weliswaar gewist maar ook bewaard. Wordt dezelfde opdracht met dezelfde waarden van *x* en *y* herhaald, dan wordt de oorspronkelijke toestand weer hersteld. In het gegeven programma wordt het cirkelschijfje in kleine stapjes van links naar rechts bewogen. De oude positie wordt telkens met een PUT-opdracht gewist en de oorspronkelijke situatie wordt hersteld, vervolgens wordt het schijfje, eveneens met PUT, op een nieuwe positie neergezet. Voor de duidelijkheid zijn op het scherm ook nog twee rechte lijnen aangebracht.

```
REM ***principe computeranimatie***
REM ***naam:ANIMA***
   SCREEN 12 : CLS
   DIM FIGURE%(496)
   CIRCLE (40,40),20,4 : PAINT (40,40),4
   LINE (0,240)-(639,280) : LINE (0,280)-(639,240)
   GET (20,20)-(60,60),FIGURE%
   PUT (120,240),FIGURE%
   FOR I=0 TO 100
      X=120+4*I : Y=240
      PUT (X,Y),FIGURE% : PUT (X+4,Y),FIGURE%
      A$=INPUT$(1)
   NEXT I
END
```

Het programma ANIMA is alleen bedoeld als illustratie van het principe van computeranimatie. Veel aardiger is het programma STARMOVE waarin een sterretje over een Lissajous-kromme over het scherm beweegt. Het is geen echte animatie, omdat de vorige posities niet gewist worden, tenminste niet meteen. In een zevental regels wordt het sterretje, een stervormige regelmatige zevenhoek, gedefinieerd. De bewe-

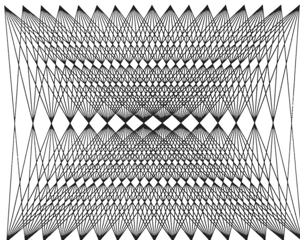
ging van het sterretje gaat door totdat op een willekeurige toets gedrukt wordt.

```
REM ***bewegende sterren***
REM ***naam:STARMOVE***
   SCREEN 12 : CLS
   DIM X(7),Y(7),STAR%(496)
   XM=320 : YM=240 : PI=4*ATN(1)
REM ***definitie van ster***
   FOR K=0 TO 6
      X(K)=40+16*COS(2*K*PI/7)
      Y(K)=40+12*SIN(2*K*PI/7)
   NEXT K
   FOR K=0 TO 6 : L=(K+4) MOD 7
      LINE (X(K),Y(K))-(X(L),Y(L)),14
   NEXT K
   GET (20,20)-(60,60),STAR% : CLS : K=0
REM ***hoofdprogramma***
   DO WHILE INKEY$="" AND K<2000
      X=XM+240*COS(K*PI/25) : Y=YM+180*SIN(K*PI/30)
      PUT (X,Y),STAR% : PUT (2*XM-X,Y),STAR%
      K=K+1
   LOOP
END
```

3 Rechte en kromme lijnen

3.1 Weefsels van rechte lijnen

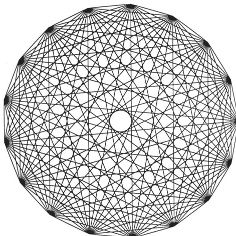
De volgende serie programma's heeft veel te maken met rechte lijnen. Hierbij valt te denken aan een stervormige figuur gevormd door de diagonalen van een regelmatige veelhoek of aan een bewegende lijn in een opeenvolging van standen. De programma's kunnen kort zijn. Om een rechte lijn af te beelden moeten we beschikken over de coördinaten van het beginpunt P_1 en van het eindpunt P_2. In het eerste programma WEB nemen we voor P_1 een aantal punten aan de bovenzijde van het scherm en voor P_2 op analoge wijze een rijtje aan de onderkant. Wordt het coördinatenstelsel met bijvoorbeeld de opdracht WINDOW (-.2,-1.2)-(.2,1.2) bepaald, dan kunnen we de coördinaten van P_1 beschrijven als $x_1 = i/n$, $y = 1$, waarbij n een niet te groot geheel getal is en i een telindex die loopt van 0 tot en met n. Er zijn dus in totaal $n + 1$ punten. Aan de onderkant ligt een overeenkomstig rijtje met $x_2 = j/n$, $y = -1$. De bedoeling van het programma is om in principe alle lijnen P_1P_2 te tekenen. Dat kan heel eenvoudig door de opdracht LINE (X1,Y1)-(X2,Y2) in een dubbele lus te plaatsen. Leuker is om het programma nog iets algemener te maken door bijvoorbeeld niet alle mogelijke verbindingslijnen af te beelden en door kleuren te gebruiken. In het gegeven programma wordt dat bereikt door de LINE-opdracht afhankelijk te stellen van de waarde $|i-j|$ die uit de twee telindices afgeleid kan worden. Er wordt alleen een lijn afgebeeld wanneer dat getal oneven is en verder wordt diezelfde waarde gebruikt als kleurattribuut. In verband daarmee is n op de vaste waarde 16 gezet. Het programma kan met weinig moeite gevarieerd worden. Overigens worden plaatjes niet altijd mooier naarmate er meer kleuren gebruikt worden.



Figuur 3.1 Een web van lijnen

```
REM ***netwerk van lijnen***
REM ***naam:WEB***
    SCREEN 12 : CLS
    WINDOW (-.2,-1.2)-(1.2,1.2)
    N=16
    FOR I=0 TO N
        FOR J=0 TO N
            C=ABS(I-J)
            X1=I/N : Y1=1 : X2=J/N : Y2=-1
            IF C MOD 2 = 1 THEN LINE (X1,Y1)-(X2,Y2),C
        NEXT J
    NEXT I
END
```

In het vorige hoofdstuk hebt u gezien dat de punten op de omtrek van een cirkel met een straal r bepaald kunnen worden door $x = r\cos(\alpha)$ en $y = r\sin(\alpha)$, waarbij α een hoek is in bijvoorbeeld het interval $[0,2\pi]$. De punten waarvoor geldt $\alpha = 2k\pi/n$, waarbij n een vast geheel getal is en k een rangnummer is dat van 1 tot n loopt, vormen met elkaar de hoekpunten van een regelmatige n-hoek. Verbinden we – zoals in het programma KANT – elk hoekpunt met elk ander hoekpunt, dan ontstaat een fraaie stervormige figuur die doet denken aan een kanten kleedje. Het aantal hoekpunten is gekozen als 17, er zijn dan 136 verbindingslijnen, 17 zijden en 119 diagonalen.

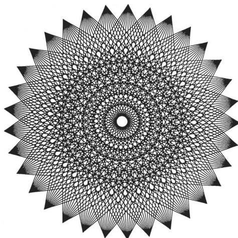


Figuur 3.2 Een regelmatige 17-hoek met alle diagonalen

```
REM ***regelmatige veelhoek met alle diagonalen***
REM ***naam:KANT***
    SCREEN 12 : CLS : PI=4*ATN(1)
    WINDOW (-1.6,-1.2)-(1.6,1.2)
    N=17 : DIM X(N),Y(N)  'aantal zijden
    FOR K=1 TO N
        X(K)=COS(2*K*PI/N)
        Y(K)=SIN(2*K*PI/N)
    NEXT K
    FOR I=1 TO N
        FOR J=1 TO I-1
            LINE (X(I),Y(I))-(X(J),Y(J))
        NEXT J
    NEXT I
END
```

Met een beetje meer werk is het mogelijk van een bepaalde regelmatige veelhoek een meer beperkte serie diagonalen af te beelden. Nemen we bijvoorbeeld een regelmatige 33-hoek en kiezen we het hoekpunt met rangnummer 1, dan zijn er achtereenvolgens zijden van het type 1-2, de kleinste diagonalen van het type 1-3, de diagonalen van het type 1-4, 1-5, ... en ten slotte de langste diagonaal, tegelijkertijd een middellijn van het type 1-17. In het programma STERPQ kiezen we twee getallen p en q als definitie van de kleinste te tekenen diagonaal en grootste

diagonaal. Het getal p vertegenwoordigt dan een diagonaal waarbij p - 1 hoekpunten worden overgeslagen. Bij elke diagonaal worden tegelijkertijd alle andere van hetzelfde type afgebeeld. Het resultaat is wederom een stervormige figuur. Het programma is met wat tekst aangekleed met de mogelijkheid de getallen p en q extern te definiëren. Ook het getal n kan aan het begin ingevoerd worden. De eigenlijke kern van het programma is wederom heel kort.



Figuur 3.3 Een voorbeeld van een sterveelhoek

```
REM ***sterveelhoek met enkele diagonalen***
REM ***naam:STERPQ***
    SCREEN 12 : CLS : PI=4*ATN(1)
    DIM X(100),Y(100)
    WINDOW (-1.6,-1.2)-(1.6,1.2)
REM ***input data***
    PRINT"kies aantal zijden n, neem n<100"
    PRINT"een goede keuze is 33"
    INPUT"aantal zijden = ",N : CLS
    PRINT"alle diagonalen orde p tot aan q worden afgebeeld"
    PRINT"orde 1 is zijde, orde 2 is diagonaal met overslaan"
    PRINT"van een hoekpunt, orde p is overslaan p-1 hoekpunten"
    PRINT"geef waarden van p en van q, een goede combinatie is"
    PRINT"n=33, p=11, q=17" : PRINT : PRINT
    INPUT"p = ",P : INPUT"q = ",Q : CLS
    IF P>Q THEN SWAP P,Q
```

```
REM ***graphics***
    FOR K=0 TO N-1
        X(K)=COS(2*K*PI/N)
        Y(K)=SIN(2*K*PI/N)
    NEXT K
    FOR I=P TO Q : FOR J=0 TO N-1
        K=(I+J) MOD N
        LINE (X(J),Y(J))-(X(K),Y(K))
    NEXT J : NEXT I
END
```

3.2 Omhullenden

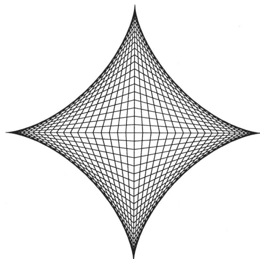
In de volgende serie programma's laten we een door twee eindpunten P_1 en P_2 gedefinieerde lijn een beweging uitvoeren.

In de wiskundige beschrijving hangen de coördinaten af van een parameter t waaraan we de betekenis van tijd kunnen hechten. Uiteraard moeten we bij het rekenwerk de tijd kleine sprongen laten maken om te vermijden dat de figuur te vol wordt met lijnen. In het programma ASTROIDE loopt P_1 heen en weer langs de x-as. Het andere eindpunt P_2 loopt op analoge wijze over de y-as. De wiskundige beschrijving is:

$$x_1 = \cos(t) \qquad y_1 = 0$$
$$x_2 = 0 \qquad y_2 = \sin(t)$$

Daarbij loopt t van 0 tot 2π in een willekeurig aantal, zeg n, stapjes. We stellen bijvoorbeeld $t = 2\pi i/n$ en we laten de teleenheid i van 0 tot n-1 lopen. De waarde $i = n$ is niet nodig, omdat dan dezelfde lijn ontstaat als bij $i = 0$. Kiezen we veel lijnen, bijvoorbeeld 64, dan ontstaat een soort ster met vier punten, een zogenaamde astroïde. Die kromme wordt hier niet gevormd uit een opeenvolging van punten, wat normaal gesproken het geval is, maar als een object dat *omhuld* wordt door lijnen, een zogenaamde *omhullende* (in het Engels: envelope).

```
REM ***astroide***
REM ***naam:ASTROIDE***
    SCREEN 12 : CLS : PI=4*ATN(1)
    WINDOW (-1.6,-1.2)-(1.6,1.2)
    LOCATE 1,1 : INPUT"aantal lijnen tussen 32 en 128 = ",N
    N=2*INT(N/2) : IF N<32 THEN N=64
    LOCATE 1,1 : PRINT SPACE$(80)
    FOR I=0 TO N-1
        T=2*PI*I/N
        LINE (COS(T),0)-(0,SIN(T))
    NEXT I
END
```



Figuur 3.4 De astroïde

Van veel belang in de techniek zijn de zogenaamde cycloïdale krommen. Ze kunnen gedefinieerd worden als zogenaamde rolkrommen, het resultaat van een samengestelde beweging van twee cirkelschijven (zie het vorige hoofdstuk). In dit geval ontstaan ze als omhullenden van de rechte lijnen P_1P_2, waarbij zowel P_1 als P_2 over dezelfde cirkel lopen. In het programma CYCLO1 hebben P_1 en P_2 de coördinaten (a_1,b_1) en (a_1,b_2). We laten P_1 en P_2 met verschillende snelheden over de cirkelomtrek, in dit geval de eenheidscirkel, lopen. In het bijzonder zorgen we ervoor dat na een beperkt aantal omlopen zowel P_1 als P_2 weer hun oorspronkelijke positie innemen. Als voorbeeld is de verhouding 1:3 gekozen. Het programma zou nu snel geschreven kunnen worden op de wijze van het vorige programma, maar er is een complicatie. De kans bestaat dat P_1 en P_2 tijdens hun rondgang ergens dezelfde positie innemen en dus samenvallen. De lijn P_1P_2 is dan niet gedefinieerd en de computer geeft een foutmelding. Om dat te voorkomen is een simpele maar doeltreffende truc toegepast. We verschuiven P_2 een heel klein beetje, zo weinig dat we er grafisch niets van merken. Dat gebeurt met het getal EPS (in de wiskunde is de Griekse letter epsilon traditioneel een heel klein getal). In het programma is nog een verfraaiing aangebracht door de lijnstukken P_1P_2 te verlengen tot aan de rand van het eenheidsvierkant, het vierkant

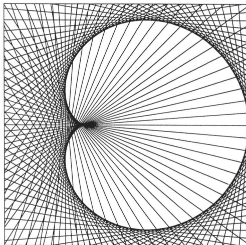
dat de eenheidscirkel omgeeft. Dit noodzaakt ons een kleine berekening uit te voeren om vast te stellen welke zijden van dat vierkant waar gesneden worden. Die berekening wordt uitgevoerd in een viertal regels. Wie voldoende kennis heeft van de coördinatenmeetkunde, kan de wiskundige betekenis daarvan gemakkelijk nagaan.

```
REM ***cycloidale kromme***
REM ***naam:CYCLO1***
    SCREEN 12 : CLS
    PI=4*ATN(1) : EPS=.0001
    WINDOW (-1.6,-1.2)-(1.6,1.2)
        K=1 : L=3 : A=.8 : N=80
    LINE (-1,-1)-(1,1),,B
    FOR J=0 TO N : T=2*PI*L*J/N
        A1=COS(T) : B1=SIN(T)
        A2=COS(K*T/L) : B2=SIN(K*T/L+EPS)
        P=B1-B2 : Q=A1-A2 : R=(A1*B2-A2*B1)*A  : S=1
        IF ABS(Q-R)<=ABS(P) THEN U(S)=(Q-R)/P : V(S)=-1 : S=S+1
        IF ABS(Q+R)<=ABS(P) THEN U(S)=-(Q+R)/P : V(S)=1 : S=S+1
        IF ABS(P-R)<ABS(Q) THEN U(S)=-1 : V(S)=(P-R)/Q : S=S+1
        IF ABS(P+R)<ABS(Q) THEN U(S)=1 : V(S)=-(P+R)/Q : S=S+1
        LINE (U(1),V(1))-(U(2),V(2))
    NEXT J
END
```

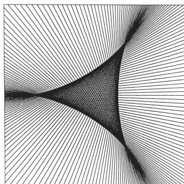
In het programma CYCLO1 kunt u zelf via de 'edit' procedure veranderingen aanbrengen door andere waarden voor *k*, *l*, *a* en *n* te nemen. Het aantal lijnen wordt bepaald door *n*. Neemt u *n* in de orde van enige tientallen, 50 of meer, dan ontstaan in de regel de beste plaatjes. Met *k* en *l* kiest u het type cycloïde, de parameter *a* is een soort schaalfactor. Het volgende programma CYCLO2 is een demonstratieversie van het vorige programma waarbij drie verschillende soorten cycloïden getoond worden. In figuur 3.5 is de eenvoudigste cycloïde afgebeeld, de zogenaamde hartlijn of cardioïde. Diezelfde lijn komt ook te voorschijn bij de alom bekende figuur van Mandelbrot. In figuur 3.6 ziet u de zogenaamde hypocycloïde van Steiner, genoemd naar een beroemde Zwitserse meetkundige uit de vorige eeuw. Let in het programma ook op de VIEW-opdracht, een manier om de te tekenen figuur in een apart gekleurd kader op het scherm te zetten.

```
REM ***cycloidale  krommen***
REM ***naam:CYCLO2***
    SCREEN 12 : CLS : PI=4*ATN(1)
    N=100  'aantal lijnen
    FOR J=1 TO 3
        VIEW (120,40)-(520,440),1,15
        WINDOW (-1,-1)-(1,1)
        SELECT CASE J
```

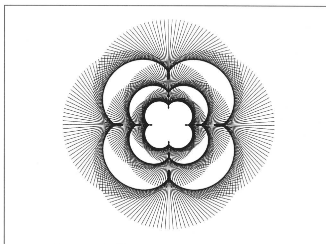
```
        CASE 1
           K=1  :  L=2  :  A=1
        CASE 2
           K=-1  :  L=2  :  A=.25
        CASE 3
           K=1  :  L=3  :  A=.8
        END SELECT
REM ***graphics***
    FOR I=1 TO N : T=2*PI*L*I/N
        A1=COS(T)  :  B1=SIN(T)
        A2=COS(K*T/L+.001)  :  B2=SIN(K*T/L+.001)
        P=B1-B2  :  Q=A1-A2  :  R=(A1*B2-A2*B1)*A   :  S=1
        IF ABS(Q-R)<=ABS(P)  THEN  U(S)=(Q-R)/P  :  V(S)=-1  :  S=S+1
        IF ABS(Q+R)<=ABS(P)  THEN  U(S)=-(Q+R)/P  :  V(S)=1  :  S=S+1
        IF ABS(P-R)<ABS(Q)  THEN  U(S)=-1  :  V(S)=(P-R)/Q  :  S=S+1
        IF ABS(P+R)<ABS(Q)  THEN  U(S)=1  :  V(S)=-(P+R)/Q  :  S=S+1
        LINE  (U(1),V(1))-(U(2),V(2))
    NEXT I : GOSUB wacht
    NEXT J : BEEP
END
wacht:
    T=TIMER
    DO
    LOOP WHILE TIMER-T<3
RETURN
```



Figuur 3.5 Een cardioïde of hartlijn



Figuur 3.6 De hypocycloïde van Steiner



Figuur 3 .7 Een stroboscoop van een draaiend lijnstuk

3.3 Draaiende lijnen

In figuur 3.7 is een grafiek afgebeeld die berust op het draaien van een lijnstuk AB van een gegeven lengte, een stok dus. Wanneer we een over-

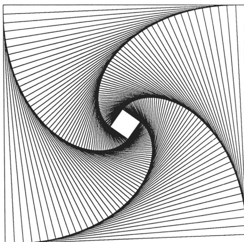
eenkomstig fysisch experiment zouden uitvoeren, voert de stok een continue beweging uit. We stellen ons voor dat we er een film van maken of dat we er met een stroboscoop naar kijken. We zien dan dat de doorlopende beweging is vervangen door een discrete, schokkerige, beweging waarbij de opeenvolgende posities onderling weinig verschillen. In het programma TURNLINE laten we het midden P_0 van AB over een cirkel bewegen.

Ten opzichte van P_0 laten we de uiteinden van de stok eveneens een cirkelbeweging uitvoeren, maar met een driemaal zo grote snelheid. Na een rondgang is al een aardig figuurtje ontstaan. Wanneer we het geheel nog een paar keer herhalen met telkens een vaste verkleining, wordt het resultaat nog leuker. Het programma past overigens ook heel goed bij het overeenkomstige programma CIRKELB in het vorige hoofdstuk. Het programma kan gemakkelijk gevarieerd worden door aan de parameters *a*, *b*, *r* en *n* andere waarden toe te wijzen.

```
REM ***patroon gevormd door een draaiende lijn***
REM ***naam:TURNLINE***
    SCREEN 12 : CLS
    WINDOW (-2,-1.5)-(2,1.5)
    A=.4 : B=6 : R=.6 : N=256
    PI=4*ATN(1) : M=3
    FOR K=0 TO M-1
        FOR L=0 TO N-1
            RK=R^K : T1=2*L*PI/N : T2=B*L*PI/N
            X0=RK*COS(T1) : Y0=RK*SIN(T1)
            X1=A*RK*COS(T2) : Y1=A*RK*SIN(T2)
            LINE (X0-X1,Y0-Y1)-(X0+X1,Y0+Y1)
        NEXT L
    NEXT K
END
```

Om interessante plaatjes te maken is enige meetkundige kennis onontbeerlijk, tenminste bij het ontwerpen van een programma. Om een programma te begrijpen en de betekenis van de verschillende opdrachten te doorzien is minder inspanning nodig wanneer men bereid is bepaalde berekeningen voor lief te nemen. Dat geldt met name voor het volgende programma WERVEL. We beginnen daarin met bijvoorbeeld een vierkant. Dat vierkant wordt ten opzichte van het middelpunt over een kleine hoek ß (in het programma b) gedraaid en tegelijk enigszins verkleind op een zodanige wijze dat de hoekpunten van het nieuwe vierkant op de zijden van het oorspronkelijke vierkant liggen. De verkortingsfactor kunnen we uitrekenen en die wordt in het programma met c aangeduid. De procedure wordt vele malen herhaald zodat we ten slotte een wervel zien van in elkaar passende vierkanten. Wat lukt voor een vierkant kan

voor elke regelmatige veelhoek uitgevoerd worden. Het aantal zijden *p* kan aan het begin gekozen worden en het programma doet de rest.



Figuur 3.8 Een vierkante wervel

```
REM ***een draaiende en krimpende veelhoek***
REM ***naam:WERVEL***
   SCREEN 12 : CLS : PI=4*ATN(1)
   WINDOW (-1.6,-1.2)-(1.6,1.2)
   INPUT"aantal zijden van veelhoek= ",P
   CLS : DIM X(P),Y(P)
   B=.05  'rotatiehoek in radialen
   A=PI*(1-2/P) : C=SIN(A)/(SIN(B)+SIN(A+B))
   FOR K=0 TO P : T=(2*K+1)*PI/P
      X(K)=SIN(T) : Y(K)=COS(T)
   NEXT K
   FOR N=1 TO 64  : PSET (X(0),Y(0))
      FOR L=1 TO P : LINE -(X(L),Y(L))
      NEXT L
      FOR M=0 TO P : Z=X(M)
         X(M)=(X(M)*COS(B)-Y(M)*SIN(B))*C
         Y(M)=(Z*SIN(B)+Y(M)*COS(B))*C
      NEXT M
   NEXT N
END
```

Minder ingewikkeld is het demonstratieprogramma OMHUL waarin van een lijnstuk P_1P_2 de beide uiteinden over een Lissajous-kromme $x = \sin(t)$, $y = \sin(at+c)$ lopen. Na een aanloopje worden de vroegere posities van de lijn weer gewist, hetgeen op het scherm de indruk wekt van een bewegende waaier van lijnen. Om het wissen van eens getekende lijnen goed te laten verlopen is doelbewust gebruik gemaakt van pixelcoördinaten en van de desbetreffende schermresolutie. Daarbij is het belangrijk dat de berekening met dubbele precisie uitgevoerd wordt en dat de coördinaten van de eindpunten van de bewegende lijn integers zijn.

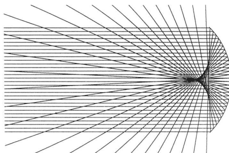
```
REM ***doorlopende omhullende***
REM ***naam:OMHUL***
    SCREEN 12 : CLS
    WINDOW (-320,-240)-(319,239)
    DEFDBL A-T : DEFINT X,Y
    F=200 : A=1.35 : T0=2
    H=.01 : C=.5 : T=0
    DO WHILE T<100 AND INKEY$=""
        X0=F*SIN(T-T0) : Y0=F*SIN(A*(T-T0)+C)
        X1=F*SIN(T) : Y1=F*SIN(A*T+C)
        X2=F*SIN(T+T0) : Y2=F*SIN(A*(T+T0)+C)
        LINE (X0,Y0)-(X1,Y1),0
        LINE (X1,Y1)-(X2,Y2)
        T=T+H
    LOOP
END
```

3.4 Een holle spiegel

Een aardige praktische toepassing van de in dit hoofdstuk besproken methode is de figuur die gevormd wordt door spiegeling van een bundel evenwijdige lichtstralen die invallen op een cilindrische holle spiegel. U kunt het verschijnsel zelf waarnemen bij het drinken van een kop thee bij het licht van een lage lichtbron. De gereflecteerde stralen gaan bij benadering door een brandpunt, maar wanneer u beter kijkt, ziet u dat ze een bepaalde kromme omhullen, een zogenaamde brandkromme (in het Engels: caustic). In het programma CAUSTIC is de simulatie uitgevoerd voor 20 evenwijdige lichtstralen. Het programma vertoont programmatechnisch niets nieuws, maar er is natuurlijk wel enige meetkundige kennis mee gemoeid.

```
REM ***brandkromme gevormd door een holle spiegel***
REM ***naam:CAUSTIC***
    SCREEN 12 : CLS
    VIEW SCREEN (0,80)-(639,400)
    WINDOW (-2.5,-1.5)-(1.5,1.5)
```

```
   N=20  'aantal stralen
REM ***spiegel***
   PSET (COS(.78),-SIN(.78))
   FOR J=-50 TO 50 : S=.78*J/50
      LINE -(COS(S),SIN(S))
   NEXT J
   FOR K=1 TO N
       B=(2*K-N-1)*.695/(N-1)
       A=SQR(1-B*B) : C=2*A*B /(1-2*B*B)
       U1=A-(B-1.5)/C : U2=A-(B+1.5)/C : V=B-(A+2)*C
       LINE (-2,B)-(A,B),14 'ingaande straal
REM ***spiegeling***
       IF V>1.5 THEN
          LINE -(U1,1.5),14
       ELSEIF V<-1.5 THEN
          LINE -(U2,-1.5),14
       ELSE
          LINE -(-2,V),14
       END IF
       T=TIMER : DO : LOOP WHILE TIMER-T<1
   NEXT K : BEEP
END
```



Figuur 3.9 Een brandlijn gevormd bij een holle spiegel

3.5 Een kaleidoscoop

Dit hoofdstuk wordt afgesloten met het eenvoudige demonstratieprogramma KALEIDO waarbij telkens een toevallig gekozen groepje gekleurde lijnen, in een symmetrische positie over het scherm verdeeld, te voorschijn komt en later weer gewist wordt. Het programma spreekt voor zich. Elke groep bestaat uit acht posities met de symmetrie van een vierkant met zijn diagonalen. Na een aanloop waarin 10 groepen van

lijnen gedefinieerd worden, volgt de hoofdlus die net zolang herhaald wordt totdat u er genoeg van hebt. De kleuren worden eveneens volgens toeval bepaald met het attribuut 1 + int(14*rnd), alle kleuren minus zwart en helwit. Het attribuut 0 wordt gebruikt om eerder getekende lijnen te wissen.

```
REM ***een kaleidoscoop van gekleurde lijnen***
REM ***naam:KALEIDO***
    SCREEN 12 : CLS
    VIEW SCREEN (40,30)-(600,450),0,15
    RANDOMIZE 111
    WINDOW (-1,-1)-(1,1)
    LOCATE 1,7 : PRINT"start/stop het programma door";
    PRINT" het drukken van een willekeurige toets"
    A$=INPUT$(1) : LOCATE 1,1 : PRINT SPACE$(80)
    DIM X1(10),Y1(10),X2(10),Y2(10)
    FOR I=1 TO 10
        X1(I)=2*RND-1 : Y1(I)=2*RND-1
        X2(I)=2*RND-1 : Y2(I)=2*RND-1
        X1=X1(I) : Y1=Y1(I) : X2=X2(I) : Y2=Y2(I)
        COL=1+INT(14*RND)
        GOSUB graphics
    NEXT I
    DO WHILE INKEY$=""
        FOR J=1 TO 10
            X1=X1(J) : Y1=Y1(J) : X2=X2(J) : Y2=Y2(J) : COL=0
            GOSUB graphics
            X1(J)=2*RND-1 : Y1(J)=2*RND-1
            X2(J)=2*RND-1 : Y2(J)=2*RND-1
            X1=X1(J) : Y1=Y1(J) : X2=X2(J) : Y2=Y2(J)
            COL=1+INT(14*RND)
            GOSUB graphics
        NEXT J
    LOOP
END
graphics:
    LINE (X1,Y1)-(X2,Y2),COL
    LINE (X1,-Y1)-(X2,-Y2),COL
    LINE (-X1,Y1)-(-X2,Y2),COL
    LINE (-X1,-Y1)-(-X2,-Y2),COL
    LINE (Y1,X1)-(Y2,X2),COL
    LINE (Y1,-X1)-(Y2,-X2),COL
    LINE (-Y1,X1)-(-Y2,X2),COL
    LINE (-Y1,-X1)-(-Y2,-X2),COL
RETURN
```

4 Computerkunst

4.1 Inleiding

Speelse grafische patronen die zijn ontworpen met behulp van een computer en ons op de een of andere wijze beroeren, kan men objecten van computerkunst noemen. We wagen ons niet aan een definitie van kunst, die bestaat óok niet of is alleen van heel beperkte waarde. De opvattingen over wat kunst is of hoort te zijn lopen sterk uiteen. Wat de een mooi vindt, ervaart een ander als lelijk, orde voor de een is chaos voor de ander. In dit hoofdstuk beperken we ons trouwens tot wat gewoonlijk als *toegepaste kunst* of *decoratieve kunst* wordt beschouwd. Het voornaamste is het ontwerpen van een vlakversiering die kan dienen als bijvoorbeeld een wandversiering, behang of een tegeltableau, een parketvloer of een tapijt. Ons visuele zintuig wordt vaak aangenaam getroffen door een herhaling van hetzelfde motief. Het is blijkbaar prettig iets bekends op een andere plaats opnieuw tegen te komen. En aldus placht de mens reeds in het verre verleden potten en schotels te versieren met herhalende elementen, zoals vissen, vogels en bloemen, en meer abstract ook de meanderrand. Op analoge wijze werden muren van bijzondere gebouwen voorzien van fraaie tegels, het Spaanse Alhambra maar ook het Hollandse binnenhuis. Escher en anderen hebben daaruit inspiratie geput voor hun eigen werk in deze moderne tijd. Herhaling in de zin van symmetrie is kennelijk een belangrijk ingrediënt van de toegepaste kunst. Maar ook het onverwachte kan een belangrijk element zijn. Een te ver doorgevoerde regelmaat kan tot een soort saaiheid en zelfs tot verveling leiden. Daarom kan een op de juiste wijze toegepast element van *toeval* in de toegepaste kunst heel belangrijk zijn.

In dit hoofdstuk bespreken we een paar eenvoudige programma's met behulp waarvan een grote verscheidenheid aan 'artistieke' grafische voorstellingen vervaardigd wordt. Daarbij kunnen soms ook kleuren ge-

bruikt worden. De computer verlost ons van veel technische rompslomp. De laserprinter tovert een eenvoudig meetkundig plaatje om tot een wonderlijk spel van lijnen, zoals uit de bijgaande illustraties blijkt.

Het enige wat er lijkt over te blijven voor de 'computerkunstenaar' is een goede keuze te maken uit het onvoorstelbare aantal mogelijkheden. Daarbij geldt het oude adagium 'in de beperking toont zich de meester', een uitspraak die zeker op het gebruik van kleuren van toepassing is.

4.2 Pixelpatronen

Een belangrijk thema is vervat in het programma PATROON1. Stel u op het beeldscherm een onzichtbaar superschaakbord van $m*m$ velden voor, waarbij m bijvoorbeeld 100 kan zijn. Volgens een bepaalde regel worden op sommige velden puntjes geplaatst, waardoor een symmetrisch patroon ontstaat. Die regel berust op een formule en een willekeurig getal, in het programma c genoemd. De formule luidt:

$$z = (1 - x^2)\ (1 - y^2)$$

De coördinaten x en y zijn zodanig gekozen dat het vierkant correspondeert met $-1 < x < 1$ en $-1 < y < 1$. Voor elk positie binnen het vierkant, en dus voor elk veld van het schaakbord, kunnen we volgens die formule een getal z uitrekenen. Op de rand is $z = 0$ en binnen het vierkant is z positief en niet groter dan 1. Is c een willekeurig getal, bijvoorbeeld 111, dan is int(cz) een geheel getal tussen 0 en c. Dat getal gebruiken we om aan het desbetreffende veld een 'kleurwaarde' toe te kennen. In deze eenvoudige monochrome versie gaat het slechts om de keuze zwart of wit. Dat kan eenvoudig worden bereikt door bijvoorbeeld alleen voor een even waarde iets af te beelden volgens het schema:

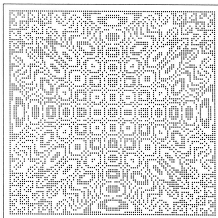
```
IF (condition) THEN PSET (x,y)
```

In het programma PATROON1 wordt standaard een vierkant van 101*101 puntjes gebruikt met een onderlinge tussenruimte van twee pixels. Het getal c kan extern ingevoerd worden, wat al meteen tot een groot aantal mogelijkheden leidt. Lage waarden van c geven een nogal saai patroon, maar naarmate c hoger is, krijgt het patroon steeds meer een toevallig karakter.

Het patroon kan op allerlei manieren gegeneraliseerd worden. Het vierkant kan een rechthoek, een cirkel of een ellips zijn en er kunnen kleuren worden gebruikt. De eenvoudigste verandering in PATROON1 is een andere uitgangsfunctie. De mogelijkheden zijn:

$z = x^2 - xy + y^2$
$z = |x| + |y| - |x + y|$

Ook de voorwaarde kan gewijzigd worden, bijvoorbeeld alleen afbeelden wanneer int(cz) een drievoud is. Een resultaat van PATROON1 is afgebeeld in figuur 4.1



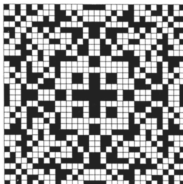
Figuur 4.1 Een symmetrisch patroon

```
REM ***een vierkant patroon***
REM ***naam:PATROON1***
   SCREEN 12 : CLS : N=50
   WINDOW (-320,-240)-(319,239)
   INPUT"geef een getal van drie cijfers ";C : CLS
   FOR I=-N TO N : FOR J=-N TO N
      IF INKEY$<>"" THEN END
      X=I/N : Y=J/N
      Z=(1-X*X)*(1-Y*Y)   'functiekeuze
      IF INT(C*Z) MOD 2 = 0 THEN PSET (3*I,-3*J)
   NEXT J : NEXT I
   LINE (-3*N-10,-3*N-10)-(3*N+10,3*N+10),,B
END
```

4.3 Tegelen en handwerken

In PATROON2 wordt op overeenkomstige wijze een soort tegelvloer afgebeeld. In plaats van puntjes verschijnen nu kleine vierkantjes. Het

programma is wat meer aangekleed en daardoor ook iets langer geworden. Maar de wiskundige kern is dezelfde als die van het vorige programma. Een van de vele resultaten is afgebeeld in figuur 4.2.



Figuur 4.2 Een tegelvloer

```
REM ***symmetrisch blokpatroon bepaald door***
REM ***een functie en een groot getal***
REM ***naam:PATROON2***
    SCREEN 12 : CLS
    GOSUB text
    WINDOW (-.5,-.3)-(1.5,1.2)
    M=32  'aantal rijen en kolommen
    GG=1000  'willekeurig groot getal
    DO
        LOCATE 1,1 : PRINT SPACE$(160) : LOCATE 1,1
        PRINT"geef een groot getal tussen 100 en 10000"
        INPUT"getal = ",GG : CLS
        FOR I=0 TO M
            LINE (0,I/M)-(1,I/M) : LINE (I/M,0)-(I/M,1)
        NEXT I
        FOR I=1 TO  M : FOR J=1 TO M
            X=-1/(2*M)+I/M : Y=-1/(2*M)+J/M
            Z=INT(GG*4*X*Y*(1-X)*(1-Y))
            IF Z MOD 2=0 THEN PAINT (X,Y)
        NEXT J : NEXT I
        LOCATE 1,1 :  PRINT"druk op R voor herhaling"
        PRINT"druk op een andere toets voor einde"
        A$=INPUT$(1)
    LOOP UNTIL A$<>"R" AND A$<>"r"
END
```

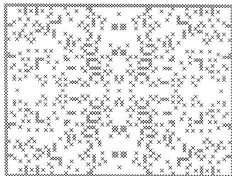
```
text:
   LOCATE 1,1
   PRINT"dit programma geeft verschillende blokpatronen"
   PRINT"na keuze van een willekeurig groot getal"
   LOCATE 4,1 : PRINT"druk op een toets ..."
   A$=INPUT$(1) : CLS
RETURN
```

Een patroon zoals afgebeeld in figuur 4.1 doet denken aan een voorbeeld in een handwerkboek. Om aan eventuele naaldkunstenaressen tegemoet te komen levert het volgende programma een kruissteekjesversie, waarbij bovendien gebruik wordt gemaakt van kleuren. Ook in dat programma kan een aantal waarden veranderd worden. Dat geldt met name voor de constante c, evenals in het vorige programma een groot getal. De toets om af te beelden is hier gericht op vijfvouden. Anders gezegd, er wordt alleen een kruisje afgebeeld wanneer int(cz) een vijfvoud is. Dat betekent een vulling van het rechthoekige kleedje met gemiddeld een vijfde van het aantal plaatsen. Uiteraard kunt u daar verandering in aanbrengen, $m = 4$ of $m = 6$ in plaats van $m = 5$. De kleuren worden op het eerste gezicht op een enigszins eigenaardige wijze bepaald. De bedoeling is om ook in de kleur een rechthoekige symmetrie te waarborgen met ongeveer dezelfde kleuren aan de korte als aan de lange kant van de rechthoek. Dat wordt bereikt met het getal L. Bij de omwerking van L tot kleurattribuut is ervoor gezorgd dat de hogere (lichtere) kleuren van 7 tot en met 15 worden gedefinieerd. Ten slotte is de rand van het borduurkleedje wit gemaakt. Dat kan natuurlijk ook allemaal anders, en voor een inventieve lezer is er nog heel wat werk te doen.

Een door PATROON3 bepaald kruissteekjespatroon is afgebeeld in figuur 4.3.

```
REM ***een kruissteekjespatroon***
REM ***naam:PATROON3***
   SCREEN 12 : CLS
   WINDOW (-320,-240)-(319,239)
   N1=40 : N2=30 : M=5
   D=6 : H=3 : B=10 : C=500
   FOR I=0 TO N1 : FOR J=0 TO N2
      IF I/N1>J/N2 THEN L=I*N2 ELSE L=J*N1
      COL=9+INT(B*L/(N1*N2)) MOD 7
      IF I=N1 OR J=N2 THEN COL=15
      X=I/N1 : Y=J/N2
      Z=(1-X*X)*(1-Y*Y)   'functiekeuze
      IF INT(C*Z) MOD M = 0 THEN GOSUB graphics
   NEXT J : NEXT I
END
graphics:
```

```
   X=D*I : Y=-D*J : GOSUB kruis
   X=D*I : Y=D*J : GOSUB kruis
   X=-D*I : Y=-D*J : GOSUB kruis
   X=-D*I : Y=D*J : GOSUB kruis
RETURN
kruis:
   PSET (X-H,Y-H),COL : LINE -(X+H,Y+H),COL
   PSET (X-H,Y+H),COL : LINE -(X+H,Y-H),COL
RETURN
```



Figuur 4.3 Een kruissteekjespatroon

Ook in het volgende programma worden kleuren gebruikt. Overeenkomstig het wiskundige principe van PATROON1 wordt de waarde van int(cz) gebruikt om het kleurattribuut vast te stellen. Zijn er bijvoorbeeld zes kleuren, dan kunnen we int(cz) modulo 6 reduceren tot een getal tussen 1 en 6 zoals in PATROON4 het geval is. De aldus afgeleide waarde hoeft niet het kleurattribuut zelf te zijn, maar kan de index zijn van een array COL(.) waarin de gekozen kleuren zijn ondergebracht.

```
REM ***een vierkant patroon met kleurenblokjes***
REM ***naam:PATROON4***
   SCREEN 12 : CLS
   WINDOW (-320,-240)-(319,239)
REM ***kleurencode***
   DIM COL(6) : N=32
   DATA 1,9,4,12,2,14
   FOR I=1 TO 6 : READ COL(I) : NEXT I
   INPUT"geef een getal van drie cijfers ";C : CLS
   LINE (-5*N-5,-5*N-5)-(5*N+5,5*N+5),,B
   FOR I=0 TO N : FOR J=0 TO N
      P1=5*I : P2=-5*I : Q1=5*J : Q2=-5*J
```

```
      X=I/N : Y=J/N
      Z=(1-X*X)*(1-Y*Y)   'functiekeuze
      L=1+INT(C*Z) MOD 6 : GOSUB graphics
   NEXT J : NEXT I
END
graphics:
   W=COL(L)
   LINE (P1-2,Q1-2)-(P1+2,Q1+2),W,BF
   LINE (P1-2,Q2-2)-(P1+2,Q2+2),W,BF
   LINE (P2-2,Q1-2)-(P2+2,Q1+2),W,BF
   LINE (P2-2,Q2-2)-(P2+2,Q2+2),W,BF
RETURN
```

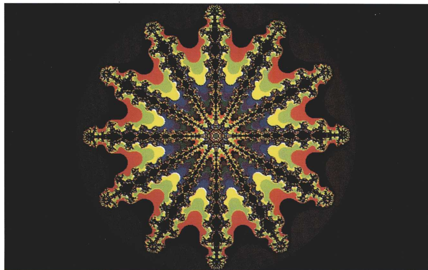
4.4 Toeval als artistieke factor

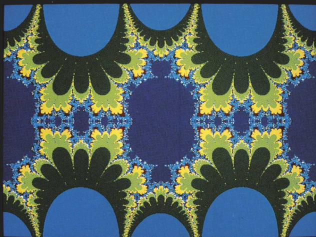
In de inleiding is al opgemerkt dat een element van toeval en onvoorspelbaarheid kan bijdragen tot een artistiek resultaat. In het programma ARTBLOK gaan we weer uit van een schaakbord van $m*m$ velden. Hier beleggen we een veld, op toevallige wijze gekozen, met een vierkant waarvan de kleur ook volgens toeval bepaald is. Wanneer een groot aantal velden aan de beurt gekomen is, kan men in het kleurige resultaat de kleuren op het beeldscherm nog op een willekeurige wijze variëren. Als extra decoratief element kunnen de dekkende vierkanten groter dan de velden van het schaakbord genomen worden, met $a = 1.3$, dus 30% groter. Het effect is dat de vierkanten van de aangrenzende velden elkaar overlappen. Dat geeft op het scherm een leuk effect dat u nog kunt beïnvloeden door voor a een andere waarde te kiezen. Omdat in het programma de opdracht PALETTE voorkomt, moet u erop letten welke Basic-variant gebruikt wordt. We geven daarom twee programma's waarbij de toevoeging Q aan de naam van het programma (ARTBLOKQ) wijst op het gebruik van Q(uick)Basic. Het programma ARTBLOK geldt voor zowel Turbo Basic als Power Basic. Wel kan het programma in Q(uick)Basic gebruikt worden wanneer u SCREEN 12 vervangt door SCREEN 9.

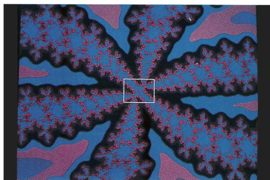
```
REM ***symmetrisch schaakbord patroon***
REM ***met door toeval bepaalde kleuren***
REM ***neem SCREEN 9 bij Q(uick)Basic***
REM ***naam:ARTBLOK***
   SCREEN 12 : CLS
   WINDOW (-.5,-.3)-(1.5,1.2)
   RANDOMIZE 111
   M=16  'aantal rijen en kolommen
   A=1.3 'grootte blokken
   KMAX=3*M*M : P=A/(2*M)
   NUMCOL=8  'aantal kleuren
```

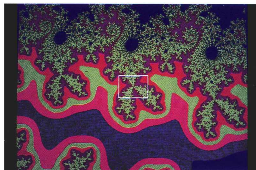
```
    GOSUB kleur : K=0
    DO WHILE INKEY$="" AND K<KMAX
        I=INT(M*RND) : J=INT(M*RND)
        L=1+INT(100*RND) MOD NUMCOL
        XC=(I+.5)/M : YC=(J+.5)/M
        X1=XC-P : Y1=YC-P : X2=XC+P : Y2=YC+P
        LINE (X1,Y1)-(X2,Y2),L,BF
        K=K+1
    LOOP
    LINE (-.04,-.04)-(1.04,1.04),14,B
    LOCATE 1,1 : A$=""
    PRINT"toets X for einde, andere toetsen voor kleurvariatie"
    DO WHILE A$<>"X" AND A$<>"x"
        GOSUB kleur
        A$=INPUT$(1)
    LOOP
END
kleur:
    FOR K=1 TO NUMCOL
        COL=INT(64*RND)
        PALETTE K,COL
    NEXT K
RETURN

REM ***moderne kunst, VGA/Q(uick)Basic versie ***
REM ***RGB kleuren***
REM ***naam:ARTBLOKQ***
    SCREEN 12: CLS
    WINDOW (-.5, -.2)-(1.5, 1.3)
    RANDOMIZE 111
    M = 16'aantal rijen en kolommen
    A = 1.3'grootte blokken
    KMAX = 3 * M * M: P = A / (2 * M)
    NUMCOL = 14: 'aantal kleuren
    GOSUB qbcolours: K = 0
    DO WHILE INKEY$ ="" AND K < KMAX
        L = 1 + INT(100 * RND) MOD NUMCOL
        I = 1 + INT(M * RND): J = 1 + INT(M * RND)
        X1 = I / M - P: Y1 = J / M - P:
        X2 = I / M + P: Y2 = J / M + P
        LINE (X1, Y1)-(X2, Y2), L, BF
        K = K + 1
    LOOP
    LINE (1 / M - 2 * P, 1 / M - 2 * P)-
                                (1 + 2 * P, 1 + 2 * P), 15, B
    LOCATE 1, 1: A$ =""
    PRINT "toets X for einde, andere toets voor kleurvariatie"
    DO WHILE A$ <> "X" AND A$ <> "x"
        GOSUB qbcolours
        A$ = INPUT$(1)
```





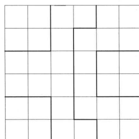




```
   LOOP
END
qbcolours:
   WIT = 40 * (1 + 256 + 65536)
   FOR K = 1 TO NUMCOL
      CR = INT(64 * RND)
      CG = INT(64 * RND)
      CB = INT(64 * RND)
      COL = CR + 256 * CG + 65536 * CB
      PALETTE K, COL
   NEXT K
   PALETTE 15, WIT
RETURN
```

4.5 Stempelen met de computer

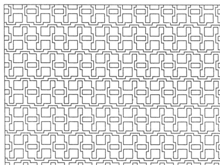
Kunstenaars hadden het vroeger maar moeilijk. Van Escher is bekend dat hij veel zorg besteedde aan het maken van houtgravures waarbij bijvoorbeeld dezelfde voorstelling vele malen in hout moest worden uitgesneden. Met een computer zou zijn produktie mogelijk verveelvoudigd zijn. Op heel bescheiden wijze kunnen we nu achter de computer zittend een 'kunstzinnig' symmetrisch patroon ontwerpen dat soms een beetje aan de werkwijze van Escher doet denken. In het programma STEMPEL stellen we ons voor dat we beschikken over een vierkant stempel, zoals afgebeeld in figuur 4.4.



Figuur 4.4 Een vierkant stempel

In dat stempel is een aantal lijnen aangebracht, vier om precies te zijn, die afgedrukt worden. Van dat stempel maken we drie kopieën zodat we over vier identieke stempels beschikken. De vier stempels, in normale of in gedraaide positie, combineren we tot een blokje van vier. Daarbij zijn

nogal wat verschillende combinaties mogelijk, in principe 4*4*4*4 of 256. Met dat blokje gaan we een rechthoekig patroon bestempelen en dat geeft een soms heel verrassend resultaat. Omdat het oorspronkelijke stempel van figuur 4.4 op een heel bijzondere wijze is gemaakt, houden de lijntjes aan de rand van het stempel niet zomaar op, maar lopen ze als het ware ononderbroken door in de overeenkomstige lijntjes van een naburige afdruk. Een mogelijk resultaat is afgebeeld in figuur 4.5.



Figuur 4.5 Een resultaat van stempelen

In het programma STEMPEL wordt de vorm van het stempel gedefinieerd door de datalijsten aan het begin en de arrays $x(.,.)$ en $y(.,.)$. Eerst wordt het stempel zelf even getoond in wit op een rood veld van ruitjes. In de volgende fase – even op een willekeurige toets drukken – wordt een groepje van vier samengesteld door vier getallen te kiezen uit 1, 2, 3 en 4. Na deze voorbereiding is het eigenlijke programma vrij kort en wiskundig doorzichtig, maar administratief vergt het toch nog het nodige detailwerk. Zoals opgemerkt, kunt u met de verschillende keuzemogelijkheden van 1, 2, 3 en 4 heel wat varianten van figuur 4.5 samenstellen. Ook kunt u op overeenkomstige wijze met een zelfgesneden stempel aan de gang gaan.

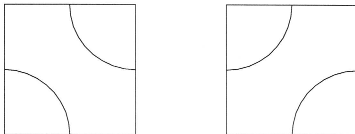
```
REM ***een regelmatig patroon van een enkel stempel***
REM ***naam:STEMPEL***
    SCREEN 12 : CLS
    DIM X(4,6),Y(4,6)
    M=4  'aantal lijnen van motief
    M(1)=3 : M(2)=3 : M(3)=4 : M(4)=6
    DATA 1,-3,1,-1,3,-1,1,3,1,1,3,1
    DATA -3,-1,-1,-1,-1,1,-3,1,-1,-3,-1,-2,0,-2,0,2,-1,2,-1,3
        FOR K=1 TO M : FOR L=1 TO M(K)
```

```
        READ X(K,L),Y(K,L)
    NEXT L : NEXT K
    LOCATE 1,1 : PRINT"tegel in normale stand"
    WINDOW (-8,-6)-(8,6)
    FOR I=-3 TO 3 : LINE (I,-3)-(I,3),4 : NEXT I
    FOR J=-3 TO 3 : LINE (-3,J)-(3,J),4 : NEXT J
    FOR K=1 TO M
        PSET (X(K,1),Y(K,1))
        FOR L=2 TO M(K)
            LINE -(X(K,L),Y(K,L))
        NEXT L
    NEXT K : A$=INPUT$(1) : CLS
    WINDOW (-18,-16)-(110,80)
REM ***een stempel kan 4 standen innemen***
REM ***1 = normaal. 2,3,4 na een paar kwartdraaien***
REM ***te herhalen patroon is een blokje van vier posities***
REM ***geselecteerd door een groepje van 4 getallen***
    PRINT"kies achtereenvolgens een getal uit 1,2,3,4"
    PRINT"een goede keus is 3,2,4,1" : PRINT
    INPUT"eerste getal = ",S(0)
    INPUT"tweede getal = ",S(1)
    INPUT"derde getal  = ",S(2)
    INPUT"vierde getal = ",S(3) : CLS
    LINE (-3,-3)-(93,69),,B
REM ***hoofdprogramma***
    FOR J=0 TO 11 : FOR I=0 TO 15
        X=6*I : Y=6*J
        IF J MOD 2 = 0 THEN W=0 ELSE W=2
        IF I MOD 2 = 1 THEN W=W+1
        S=S(W)
        IF S=1 OR S=2 THEN P=1 ELSE P=-1
        IF S=1 OR S=4 THEN Q=1 ELSE Q=-1
        IF S=1 OR S=3 THEN
            FOR K=1 TO M : PSET (X+P*X(K,1),Y+Q*Y(K,1))
            FOR L=2 TO M(K) : LINE -(X+P*X(K,L),Y+Q*Y(K,L))
            NEXT L : NEXT K
        ELSE
            FOR K=1 TO M : PSET (X+P*Y(K,1),Y+Q*X(K,1))
            FOR L=2 TO M(K) : LINE -(X+P*Y(K,L),Y+Q*X(K,L))
            NEXT L : NEXT K
        END IF
    NEXT I : NEXT J
END
```

4.6 Toevallig stempelen

In het laatste programma van deze serie komt het toeval weer aan bod. Het programma lijkt erg op het vorige. Weer gaan we stempelen, maar

ditmaal heeft het stempel de vorm die in figuur 4.6 wordt getoond. Door de symmetrische vorm van het stempel is er maar één andere positie mogelijk, eveneens afgebeeld in figuur 4.6. Met deze stempels, naar willekeur gekozen, stempelen we een vierkant helemaal vol. Ook ditmaal lopen alle inwendige lijnen, cirkelbogen, ononderbroken in elkaar over. Het resultaat (zie figuur 4.7) is een grillig patroon, als een volgens toeval ontstaan object van decoratieve kunst. Het programma is naar de uitvinder, TRUCHET, genoemd.

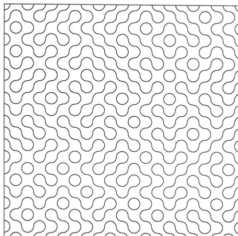


Figuur 4.6 De twee stempels van Truchet

In het programma is enige zorg besteed aan de afwerking van de cirkelvormige bogen. Dit vergt wel extra programmaregels. In Basic kan een cirkelboog ook met de opdracht CIRCLE getekend worden, maar daarbij laat de nauwkeurigheid soms te wensen over. De twee standen van het stempel zijn verwerkt in twee aparte subroutines. Om het programma korter te maken kunnen deze eventueel samengevoegd worden. In het algemeen luidt het advies niet te veel aan te sturen op het maken van 'mooie en beknopte' programma's wanneer dat ten koste van de leesbaarheid en de duidelijkheid gaat. Als extra wordt in het programma een deel van de figuur met de opdracht PAINT 'gekleurd'.

```
REM ***een moderne tegelvloer volgens Truchet***
REM ***naam:TRUCHET***
    SCREEN 12 : CLS : PI=4*ATN(1)
    WINDOW (-320,-240)-(319,239)
    RANDOMIZE 222
    R=10 : N=10 : P=2*N*R
    LINE (-P,-P)-(P,P),,B
    FOR I=-N TO N-1 : FOR J=-N TO N-1
        IF INKEY$<>"" THEN END
        X0=2*R*I+R : Y0=2*R*J+R
        IF RND<.5 THEN GOSUB aaa ELSE GOSUB bbb
    NEXT J : NEXT I
```

```
    FOR I=-N TO N : FOR J=-N TO N
        IF (I+J) MOD 2 = 0 THEN GOTO repeat
        X=2*R*I : Y=2*R*J
        IF I=-N THEN X=X+1
        IF I=N THEN X=X-1
        IF J=-N THEN Y=Y+1
        IF J=N THEN Y=Y-1
        PAINT (X,Y)
repeat:
    NEXT J : NEXT I
END
aaa:
    X=X0-R : Y=Y0-R
    FOR K=0 TO 9
        T=K*10*PI/180
        U=X+R*COS(T) : V=Y+R*SIN(T)
        IF K=0 THEN PSET (U,V) ELSE LINE -(U,V)
    NEXT K
    X=X0+R : Y=Y0+R
    FOR K=0 TO 9
        T=(180+K*10)*PI/180
        U=X+R*COS(T) : V=Y+R*SIN(T)
        IF K=0 THEN PSET (U,V) ELSE LINE -(U,V)
    NEXT K
RETURN
bbb:
    X=X0+R : Y=Y0-R
    FOR K=0 TO 9
        T=(90+K*10)*PI/180
        U=X+R*COS(T) : V=Y+R*SIN(T)
        IF K=0 THEN PSET (U,V) ELSE LINE -(U,V)
    NEXT K
    X=X0-R : Y=Y0+R
    FOR K=0 TO 9
        T=(270+K*10)*PI/180
        U=X+R*COS(T) : V=Y+R*SIN(T)
        IF K=0 THEN PSET (U,V) ELSE LINE -(U,V)
    NEXT K
RETURN
```



Figuur 4.7 Een voorbeeld van een Truchet-patroon

5 Spel van chaos en orde

5.1 Lineaire transformaties

Eerder is al gesproken over een lineaire transformatie in het platte vlak. Een dergelijke transformatie betekent meetkundig een verplaatsing van een willekeurig punt volgens een vaste formule die er in coördinaten uitziet als:

(5.1) $x' = a\,x + b\,y + e \qquad y' = c\,x + d\,y + f$

Zijn x en y de coördinaten van een willekeurig punt P, dan zijn x' en y' de coördinaten van het verplaatste punt dat we P' noemen. Gewoonlijk geven we een transformatie een naam, bijvoorbeeld T, en noteren we de verplaatsing symbolisch als $P' = TP$. Vaak onderwerpen we een punt P aan een reeks van transformaties. Noemen we de transformaties T_1, T_2, T_3, enzovoort, dan kunnen we de achtereenvolgende stappen kort en duidelijk samenvatten in de notaties $P_1 = T_1P$, $P_2 = T_2P_1$, $P_3 = T_3P_2$, enzovoort. Letten we alleen op het eindresultaat, dan schrijven we bijvoorbeeld $P_3 = (T_3T_2T_1)P$. De transformatie waardoor P_3 uit het beginpunt P afgeleid wordt, kunnen we daarbij interpreteren als een samengestelde transformatie $T_3T_2T_1$ die in de wiskunde het produkt van T_1, T_2, T_3 genoemd wordt. Bij deze notatie is de volgorde van de samenstellende factoren van belang, de factoren T moeten van rechts naar links gelezen worden.

Formules volgens het formaat (5.1) zijn veelgebruikte bouwstenen in programma's die iets met meetkunde te maken hebben. Om dergelijke programma's te begrijpen is het nuttig enig inzicht in de eigenschappen van de transformaties te hebben. Daarbij moeten we in de eerste plaats letten op punten waarbij P' met P samenvalt. Die punten kunnen we algebraïsch bepalen door in (5.1) $x' = x$ en $y' = y$ te stellen en de twee vergelijkingen naar x en y op te lossen. Meestal is er precies één op-

lossing, maar het kan voorkomen dat er helemaal geen oplossing is, of dat er oneindig veel oplossingen zijn. Het eerste is het geval bij een *translatie* of parallelverschuiving. De beschrijving in coördinaten is:

(5.2) $x' = x + e \qquad y' = y + f$

Meetkundig is dit een weinig boeiende transformatie, in feite wordt het vlak van de tekening met alles wat zich erop bevindt gewoon verschoven. Een punt dat bij de transformatie op zijn plaats blijft, noemen we een *dekpunt* (in het Engels: fixed point). Beschrijft (5.1) een meetkundige transformatie met een dekpunt en richten we ons coördinatenstelsel zo in dat het dekpunt tegelijk de oorsprong (0,0) is, dan wil dat zeggen dat in (5.1) $e = f = 0$.

De eenvoudigste transformatie van dat type is de centrale vermenigvuldiging of *schaling* waarbij de oorsprong O het centrum van de vermenigvuldiging is. De beschrijving luidt:

(5.3) $x' = s\,x \qquad y' = s\,y$

Is $s > 1$, dan gaat het om een schaalvergroting waarbij in afstand gemeten geldt $P'O = s\,PO$. Is $0 < s < 1$, dan hebben we te maken met een schaalverkleining, een zogenaamde contractie. Aan s kunnen we ook negatieve waarden toekennen. Heel speciaal is $s = -1$, meetkundig een *puntspiegeling*.

De volgende in de rij is de draaiing of *rotatie* om de oorsprong over de rotatiehoek $|\alpha|$. De wiskundige formulering luidt:

(5.4) $x' = a\,x - b\,y \qquad y' = b\,x + a\,y$

waarbij

$$a = \cos(\alpha) \text{ en } b = \sin(\alpha)$$

De draaiing vindt daarbij plaats tegen de wijzers van de klok in, in de wiskunde de positieve draaizin.

Een veelvoorkomende transformatie is de combinatie van een rotatie en een schaling. Daarbij geldt weer de beschrijving (5.4), maar nu is:

$$a = s\cos(\alpha) \quad \text{en} \quad b = s\sin(\alpha)$$

waarbij s de schaalfactor is. Deze gecombineerde transformatie heeft geen algemeen aanvaarde benaming, maar wanneer $|s| < 1$ spreken we wel van een *krimpdraaiing*. Alle opmerkingen over de coëfficiënten a, b, c, d van (5.4) gelden ook voor de algemenere vorm (5.1). Het enige verschil is dat de oorsprong geen dekpunt is.

Alle tot dusver besproken transformaties zijn zogenaamde rechtstreekse gelijkvormigheidstransformaties. Passen we die transformaties toe op een figuur in plaats van op een enkel punt, dan ontstaat een gelijkvormige figuur; een vierkant wordt een groter of kleiner vierkant in een mogelijk gedraaide positie. Bovendien blijft de draaizin bewaard, rechtsom blijft rechtsom en linksom blijft linksom. Anders is dat voor de zogenaamde indirecte gelijkvormigheidstransformaties, bewerkingen die opgebouwd zijn uit een gewone gelijkvormigheid en een spiegeling. Bij een dergelijke transformatie verandert de draaizin, een letter p gaat over in een letter q en omgekeerd. De eenvoudigste spiegeling is de *lijnspiegeling*. Een lijnspiegeling om de x-as wordt beschreven door:

$$x' = x \qquad y' = -y$$

Bij de meest algemene lineaire transformaties is de schaling niet in alle richtingen dezelfde. Een kenmerk van deze transformaties is dat een vierkant in een parallellogram overgaat en een cirkel in een ellips. Deze transformaties worden *affiene* transformaties genoemd. Hun belangrijkste meetkundige eigenschap is dat evenwijdige lijnen altijd in andere evenwijdige lijnen worden getransformeerd. Verder worden oppervlakken altijd met dezelfde factor veranderd. Die factor, de zogenaamde *oppervlaktefactor*, is gelijk aan $|ad-bc|$, de absolute waarde van de coëfficiëntendeterminant van x en y. Is $ad > bc$, dan blijft de draaizin dezelfde, maar voor $ad < bc$ gaat deze over in de tegengestelde draaizin.

Zijn P en Q twee willekeurige punten en zijn P' en Q' de getransformeerde punten, dan is de verhouding $P'Q'/PQ$ in het algemeen afhankelijk van de ligging van die punten. Dat kan in de ene richting een verkorting en in een andere richting een verlenging betekenen. Geldt echter onder alle omstandigheden $P'Q' < PQ$, dan spreken we van een verkortingstransformatie, een zogenaamde *contractie*. De bovenste grens van $P'Q'/PQ$ noemen we dan de *contractiefactor*. In de wereld van de niet-lineaire analyse met onderwerpen als chaos, vreemde aantrekkers, fractals en dynamische systemen, zijn contracties uiterst belangrijk. Reeds met een tweetal eenvoudige contracties kan een boeiend fractalpatroon vastgelegd worden, zoals u weldra zult zien.

We gaan nu meer concreet uit van twee contracties die we L en R noemen (denk maar aan Links en Rechts). In het programma FRACMC1 zijn ze gedefinieerd als formules volgens de vorm (5.1). In beide gevallen zijn het combinaties van een rotatie en een schaalverkleining, dus krimpdraaiingen. De rotatie R is praktisch identiek aan L, maar de oriëntatie is tegengesteld en het dekpunt ligt ergens anders. L heeft het dekpunt (-1,0), terwijl R het dekpunt (1,0) heeft. Uit de formules (5.4) kunnen we afleiden dat de verkortingsfactor van L gelijk is aan $\sqrt{(a^2 + b^2)}$. In de

gegeven numerieke situatie ligt dat getal ruimschoots beneden 1, zodat L en R inderdaad contracties zijn.

5.1.1 De spelregels

Na deze voorbereidingen gaan we met L en R het zogenaamde *spel van chaos* spelen. Daar is een soort dobbelsteen voor nodig, een willekeurig element in de berekeningen, iets toevalligs en onvoorspelbaars, kortom iets van chaos. De dobbelsteen wordt in de computersimulatie natuurlijk vervangen door de ingebouwde Random Number Generator (RNG), waarbij de opdracht RND telkens een 'toevallig' getal tussen 0 en 1 produceert. Met een willekeurige beginpunt P_0 als uitgangspunt wordt een in principe oneindige rij van punten P_0, P_1, P_2, ..., P_n, P_{n+1}, ... geconstrueerd waarbij elk punt uit zijn voorganger afgeleid wordt door er de transformatie L of R op toe te passen. Dus:

$$P_{n+1} = \mathrm{L}\,P_n \quad \text{of} \quad P_{n+1} = \mathrm{R}\,P_n$$

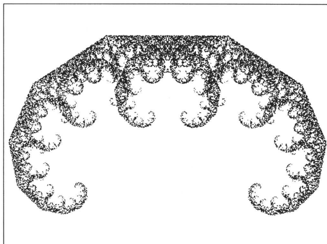
De keuze uit L of R maken we volgens een soort fifty-fifty principe door te letten op de waarde van het random getal RND, namelijk volgens het schema:

```
IF RND<.5 then L ... else R ...
```

Wanneer we de puntenrij maar lang genoeg voortzetten, ontstaat een mooi patroon dat vaak associaties oproept van bekende beelden, zoals planten, bomen en wolken. Dat patroon is een fractal, een uitspraak die nog wat precisering behoeft. In het gegeven programma is het fractalpatroon symmetrisch ten opzichte van een verticale as, en daarom kunnen we bij elk gevonden punt tegelijkertijd ook een gespiegeld punt afbeelden. Strikt genomen is de fractal F reeds wiskundig bepaald door de twee gegeven contracties L en R. Onderwerpen we F aan de contractie L, dan resulteert dit in een kleinere, gelijkvormige kopie van F die deel uitmaakt van F. Hetzelfde geldt voor R. De fractal kan opgevat worden als de limietfiguur van banen die op de bovengenoemde wijze gevormd zijn. Is P_0 geen punt van F, dan behoren ook de volgende punten van de baan niet tot F, maar ze komen er wel steeds dichter bij. Na een aantal stappen is dat overigens op het beeldscherm niet te zien. Maakt P_0 wel deel uit van F, dan liggen alle volgende punten eveneens op F. Het chaosspel betekent dan dat een punt P op toevallige wijze over de fractal heen en weer springt en, merkwaardigerwijs, op zijn reis bijna overal komt. Ondanks het feit dat de echte fractal uit oneindig veel punten bestaat, kan de fractal reeds met een beperkt aantal baanpunten op representatieve wijze afgebeeld worden. In de praktijk zijn daarvoor

vaak enige tienduizenden punten nodig, soms kan men met minder volstaan, maar voor enkele mooie plaatjes kan dat aantal in de miljoenen lopen, voor een goede computer nog steeds geen bezwaar.

5.1.2 Fractals bepaald door twee gelijkvormigheden



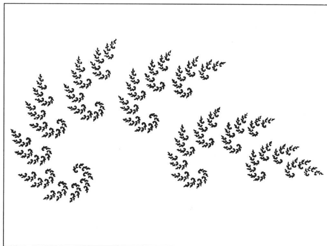
Figuur 5.1 Een fractale wolkenrand

```
REM ***iteratief systeem, twee rotaties***
REM ***symmetrische versie***
REM ***naam:FRACMC1***
   SCREEN 12 : CLS : RANDOMIZE 11
   WINDOW (-2.4,-1.2)-(2.4,2.4)
   KMAX=20000
REM ***coefficienten***
   A=.6 : B=.4 : C=A : D=-B
   X=1 : Y=0 : K=0  'start
   DO WHILE K<KMAX AND INKEY$=""
      R=RND
      IF R<.5 THEN
         X1=A*X-B*Y-1+A : Y1=B*X+A*Y+B  'rotatie (-1,0)
      ELSE
         X1=C*X-D*Y+1-C : Y1=D*X+C*Y-D  'rotatie (1,0)
      END IF
```

```
        X=X1 : Y=Y1
        PSET (X,Y) : PSET (-X,Y)
        K=K+1
    LOOP : BEEP
END
```

Het programma kan met weinig moeite gevarieerd worden door voor *a* en *b* andere getallen te nemen. Mits u ervoor zorgt dat $a^2 + b^2 < 1$, ontstaat er altijd een fractale figuur. De kans bestaat dat de fractal te groot is voor het beeldscherm, maar in dat geval kunt u het plaatje verkleinen door de WINDOW-opdracht aan te passen. Het aantal punten wordt gelimiteerd door *kmax* = 20000, maar door de INKEY$-conditie kan het programma altijd onderbroken worden wanneer de geproduceerde illustratie niet bevalt. Natuurlijk kunt u de veiligheidsmarge van *kmax* zonder probleem op een hoger getal instellen.

In het volgende programma zijn L en R weer twee rotaties, krimpdraaiingen, maar nu verschillen de contractiefactoren. Het is verstandig daarmee rekening te houden bij de keuze van L of R. Stel dat L de oppervlaktefactor 1/2 heeft en R de factor 1/4. Een vierkantje dat vijfmaal door R getransformeerd wordt, heeft tenslotte een ruim duizendmaal zo klein oppervlak ($1024 = 2^{10}$). Voor datzelfde doel zou de transformatie L tienmaal uitgevoerd moeten worden. Voor een enigszins gelijkmatige com-



Figuur 5.2 Spiraalvormig gebladerte

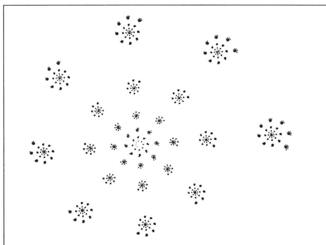
puterillustratie is het dus verstandig om L gemiddeld tweemaal zo vaak als R te kiezen (`IF RND<2/3 THEN L ... ELSE R ...`). In het programma FRACMC2 is dit op systematische wijze uitgevoerd, zodat het programma ook met andere waarden bruikbaar is.

```
EM ***iteratief systeem, twee rotaties***
REM ***naam:FRACMC2***
   SCREEN 12 : CLS : RANDOMIZE 11
   WINDOW (-2.4,-1.2)-(1.6,1.8)
   KMAX=60000
REM ***coefficienten***
   A=.6 : B=.45 : C=.5 : D=0
   DET1=A*A+B*B : DET2=C*C+D*D
   Q=DET1/(DET1+DET2)
   X=1 : Y=0  : K=0  'start
   DO WHILE K<KMAX AND INKEY$=""
      R=RND
      IF R<Q THEN
         U=A*X-B*Y-1+A : V=B*X+A*Y+B   'rotatie L
      ELSE
         U=C*X-D*Y+1-C : V=D*X+C*Y-D   'rotatie R
      END IF
      X=U : Y=V
      PSET (X,Y)
      K=K+1
   LOOP : BEEP
END
```

De keuze van de waarschijnlijkheidsdrempel q wordt in het voorgaande programma bepaald door de oppervlaktefactoren van L en R. Dat gaat in het algemeen het best wanneer de door L en R geproduceerde fractal een beetje 'oppervlakteachtig' is. Wanneer de fractal daarentegen een meer ijle structuur heeft, en meer lijnachtig is, ontstaan soms mooiere plaatjes met een andere waarde van q. In het volgende programma FRACMC3 is een keuze gemaakt door niet de oppervlaktefactoren te nemen, maar de wortels uit die factoren. Deze hebben namelijk de meetkundige betekenis van een gemiddelde verkorting van de lijnstukjes. Soms loont het de moeite om q niet automatisch door het programma te laten bepalen, maar q zelf a priori een numerieke waarde te geven. Door hetzelfde programma met telkens een andere waarde van q te herhalen, kunt u proefondervindelijk de waarde van q bepalen waarmee het plaatje het meest bevalt. Hoewel het programma FRACMC3 heel weinig afwijkt van het vorige programma, wordt het toch compleet afgedrukt.

```
REM ***iteratief systeem, twee rotaties***
REM ***naam:FRACMC3***
   SCREEN 12 : CLS : RANDOMIZE 11
   WINDOW (-3.5,-2.1)-(2.1,2.1)
```

```
   KMAX=20000
REM ***coefficienten***
   A=.65 : B=.65 : C=.1 : D=-.1
   F1=SQR(A*A+B*B) : F2=SQR(C*C+D*D)
   Q=F1/(F1+F2)
   X=1 : Y=0  : K=0  'start
   DO WHILE K<KMAX AND INKEY$=""
      R=RND
      IF R<Q THEN
         X1=A*X-B*Y-1+A : Y1=B*X+A*Y+B   'rotatie L
      ELSE
         X1=C*X-D*Y+1-C : Y1=D*X+C*Y-D   'rotatie R
      END IF
      X=X1 : Y=Y1
      PSET (X,Y)
      K=K+1
   LOOP : BEEP
END
```



Figuur 5.3 Een spiraal van sterretjes

Wie zelf wil experimenteren, dient vier getallen *a, b, c, d* te kiezen waarbij zowel $a^2 + b^2 < 1$ als $c^2 + d^2 < 1$ geldt. In tabel 5.1 is een aantal geschikte waarden opgenomen voor de programma's FRACMC2 en FRACMC3.

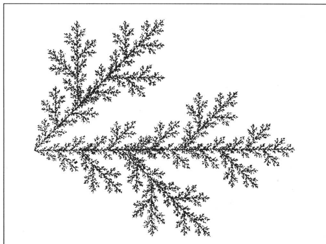
Coëfficiënten voor:

$x' = ax - by + (a\text{-}1)$, $y' = bx + ay + b$ (rotatie)
$x' = cx - dy + (1\text{-}c)$, $y' = dx + cy - d$ (rotatie)

a	b	c	d
.5	.5	.5	-.5
.5	.5	.5	.5
.6	.374	.4	-.374
.42	.3	.42	-.3
0	.707	.5	-.5
.6	.6	.53	0
.6	.45	.5	0
0	.7	.7	0
.5	.5	.6	-.2
.75	.25	.5	-.375

Tabel 5.1

De voorgaande programma's kunnen nog enigszins gevarieerd worden door in plaats van een rotatie een spiegeling te nemen. Het programma FRACMC4 geeft daarvan een voorbeeld (zie ook figuur 5.4).



Figuur 5.4 Een fractal als een bebladerde tak

```
REM ***iteratief systeem, twee spiegelingen***
REM ***naam:FRACMC4***_menu - amos   OK
   SCREEN 12 : CLS : RANDOMIZE 11
   WINDOW (-1.6,-1.2)-(1.6,1.2)
   KMAX=40000
REM ***coefficienten***
   A=.5 : B=.5 : C=.6667 : D=0
   DET1=A*A+B*B : DET2=C*C+D*D
   Q=DET1/(DET1+DET2)
   X=1 : Y=0  : K=0  'start
   DO WHILE K<KMAX AND INKEY$=""
      R=RND
      IF R<Q THEN
         X1=A*X+B*Y-1+A : Y1=B*X-A*Y+B   'spiegeling L
      ELSE
         X1=C*X+D*Y+1-C : Y1=D*X-C*Y-D   'spiegeling R
      END IF
      X=X1 : Y=Y1
      PSET (X,Y),10
      K=K+1
   LOOP : BEEP
END
```

Tabel 5.2 bevat nog een aantal waarden die voor dit programma gebruikt kunnen worden.

Coëfficiënten voor:

$x' = ax + by + (a\text{-}1)$, $y' = bx - ay + b$ (spiegeling)
$x' = cx + dy + (1\text{-}c)$, $y' = dx - cy - d$ (spiegeling)

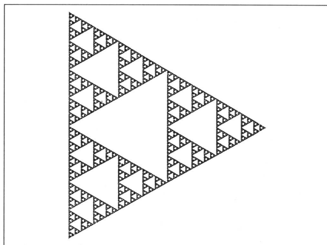
a	b	c	d
.5	.289	.5	-.289
.5	.5	.5	0
.5	.5	.667	0
.5	.5	.6	-.2
0	.64	0	-.64

Tabel 5.2

5.2 Driehoek en vierkant van Sierpinski

In plaats van twee contracties kunt u er ook drie of meer kiezen. De structuur van het programma blijft in wezen gelijk, alleen moet u bij het 'dobbelen' goed letten op de kansverdeling van de waarschijnlijkheden waarin de samenstellende contracties telkens gekozen worden. Een

klassiek voorbeeld is de *driehoek van Sierpinski*, een bekend paradepaardje in de wereld van de fractals. We gaan uit van een driehoek met de hoekpunten A, B en C. De vorm van de driehoek is niet belangrijk. Meestal wordt een gelijkzijdige driehoek of een half vierkant gekozen. De drie transformaties waarmee het chaosspel gespeeld wordt, zijn centrale vermenigvuldigingen met de factor 1/2 en de centra A, B en C. In het programma FRACMC5 is een en ander nader uitgewerkt voor een gelijkzijdige driehoek. De coëfficiënten van de drie transformaties zijn nu in een aantal arrays ondergebracht zodat u bij het dobbelproces slechts de index I hoeft te kiezen als een getal 1, 2 of 3 met gelijke kansen. In dit speciale geval voldoet het resultaat aan de symmetrie, de draaiingen en spiegelingen van een gelijkzijdige driehoek. Dat zorgt dat we bij elk gevonden punt van de baan vijf andere cadeau krijgen, dit ter verklaring van de vijf extra PSET-opdrachten. De driehoek van Sierpinski laat nog meer interpretaties toe. Sommigen zien het als een zeef met gaten, in een driehoekig stuk metaal maakt men eerst een driehoekig gat in het midden waarna er vier kleinere driehoeken, aan de hoekpunten nog samenhangend, overblijven. Uit de kleinere driehoeken worden op analoge wijze driehoekige stukjes verwijderd, enzovoort. Ten slotte blijft er een soort skelet over, de zogenaamde *zeef* van Sierpinski.



Figuur 5.5 De zeef van Sierpinski

```
REM ***de zeef van Sierpinski***
REM ***naam:FRACMC5***
   SCREEN 12 : CLS : RANDOMIZE 11
   WINDOW (-1.4,-1.2)-(1.8,1.2)
   DIM E(3),F(3)
   C=.5  'contractie factor
   A=1/2 : B=SQR(3)/2
   E(1)=1-C : F(1)=0 : E(2)=A*(C-1)
   F(2)=-B*(C-1) : E(3)=A*(C-1) : F(3)=B*(C-1)
   KMAX=10000 : K=0 : X=1 : Y=0  'start
   DO WHILE K<KMAX AND INKEY$=""
      L=1+INT(3*RND)
      X=C*X+E(L) : Y=C*Y+F(L)
      PSET (X,Y) : PSET (-A*X-B*Y,B*X-A*Y)
      PSET (-A*X+B*Y,-B*X-A*Y) : PSET (X,-Y)
      PSET (-A*X+B*Y,B*X+A*Y) : PSET (-A*X-B*Y,-B*X+A*Y)
      K=K+1
   LOOP : BEEP
END
```

Het programma kan op eenvoudige wijze gevarieerd worden door bijvoorbeeld c te wijzigen. De instelling $c = .7$ leidt tot een meetkundig object dat als een *multifractal* of *fractaldichtheid* bekend staat. Dit is een patroon dat als een zichzelf veelvuldig overlappende fractal kan worden beschouwd. Het lijkt alsof er over de driehoek zandkorreltjes gestrooid zijn die met elkaar een grillig heuvelachtig landschap vormen. In elk punt kan een soort hoogte – of liever dichtheid – gedefinieerd worden, en in een computerillustratie kan deze weergegeven worden overeenkomstig de geografische landkaart van Zwitserland. In de computersimulatie letten we op het aantal punten dat van een baan van vele tienduizenden punten in een klein vierkantje, de omgeving van een willekeurige pixel, terechtkomt. Het schema werkt als volgt. Uitgaande van een willekeurig beginpunt P_0 vormen we een baan P_0, P_1, P_2, ... door voor elk punt P_n de opvolger P_{n+1} te bepalen als TP_n waarbij T een van de vier, willekeurig gekozen, contracties is. De positie van het nieuwe punt scannen we op het scherm met de POINT-opdracht om te controleren of er al eerder een punt geplaatst is. Zo ja, dan verhogen we het kleurattribuut met een eenheid. Dit betekent dat een kleurwijziging plaatsvindt. Op die manier verandert het patroon voortdurend van kleur op het scherm, sommige delen heel sterk terwijl andere delen achterblijven. In principe kunnen we in de gewone VGA-modus maximaal tot een attribuut van 15 gaan.

Eventueel kunnen we het attribuut modulo 16 (of 15) doortellen. Ook hier bestaan weer mogelijkheden tot het aanbrengen van variaties. De kleurverandering volgt het standaard EGA-patroon zonder dat er van de PALETTE-opdracht gebruik wordt gemaakt. Het programma kan op deze wijze zowel in Q(uick)Basic als in Turbo of Power Basic uitgevoerd worden.

```
REM ***de zeef van Sierpinski met verlopende kleuren***
REM ***naam:FRACMC6***
   SCREEN 12 : CLS : RANDOMIZE 11
   WINDOW (-1.44,-1.08)-(1.44,1.08)
   DIM E(3),F(3)
   C=.7  'contractie factor
   A=1/2 : B=SQR(3)/2
   E(1)=1-C : F(1)=0 : E(2)=A*(C-1)
   F(2)=-B*(C-1) : E(3)=A*(C-1) : F(3)=B*(C-1)
   KMAX=100000 : K=0 : X=1 : Y=0  'start
   DO WHILE K<KMAX AND INKEY$=""
      L=1+INT(3*RND)
      X=C*X+E(L) : Y=C*Y+F(L)
      COL=POINT (X,Y) : COL=COL+1
      PSET (X,Y),COL : PSET(X,-Y),COL
      PSET (-A*X-B*Y,B*X-A*Y),COL
      PSET (-A*X+B*Y,-B*X-A*Y),COL
      PSET (-A*X+B*Y,B*X+A*Y),COL
      PSET (-A*X-B*Y,-B*X+A*Y),COL
      K=K+1
   LOOP : BEEP
END
```

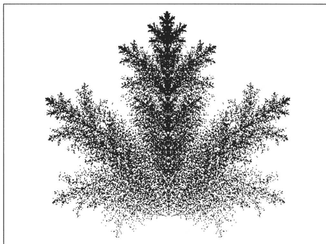
Omdat het computerscherm als het ware opgebouwd is uit een rechthoekig patroon van pixels, worden figuren met veel horizontale en verticale lijnen, met spiegelsymmetrie ten opzichte van een verticale en een horizontale as, zuiverder afgebeeld dan figuren met schuine en gebogen lijnen. Dit gegeven kan een reden zijn om van het zojuist gegeven programma een vierkante versie te maken. Het programma heeft de naam FRACMC7. Er zijn nu vier contracties, centrale vermenigvuldigingen ten opzichte van elk van de vier hoekpunten met dezelfde verkortingsfactor c.

```
REM ***Sierpinski vierkant met verlopende kleuren***
REM ***naam:FRACMC7***
   SCREEN 12 : CLS : RANDOMIZE 11
   WINDOW (-1.6,-1.2)-(1.6,1.2)
   C=SQR(1/2)  'contractiefactor
   DIM E(4),F(4)
   E(1)=1-C : F(1)=1-C
   E(2)=1-C : F(2)=-1+C
   E(3)=-1+C : F(3)=1-C
   E(4)=-1+C : F(4)=-1+C
   KMAX=100000 : K=0 : X=1 : Y=0  'start
   DO WHILE K<KMAX AND INKEY$=""
      L=1+INT(4*RND)
      X=C*X+E(L) : Y=C*Y+F(L)
      COL=POINT (X,Y) : COL=COL+1
      PSET (X,Y),COL : PSET (X,-Y),COL
```

```
        PSET (-X,Y),COL : PSET (-X,-Y),COL
        PSET (Y,X),COL : PSET (-Y,X),COL
        PSET (Y,-X),COL : PSET (-Y,-X),COL
        K=K+1
    LOOP : BEEP
END
```

5.3 Boomblad en varen

De wiskundige Michael Barnsley heeft veel aandacht besteed aan de hier besproken methode om fractals te genereren met behulp van twee of meer gelijkvormigheidstransformaties. Hij stelde zich de vraag of het ook mogelijk was de omgekeerde weg te volgen, een fractal te maken die sprekend op een gegeven object uit de ons omringende wereld lijkt, zoals een boomblad, een bloem of een vogelveer. Zijn ideeën legde hij neer in een artikel met de uitdagende titel 'Fractals op bestelling' en ook schreef hij er een bekend boek over. Helaas bevat dat boek veel wiskundige technieken en bevat het weinig bruikbare programma's. Aan zijn werk ontlenen we twee aardige voorbeelden, een blad van een boom en een varenblad. Het programma BLAD omvat drie eenvoudige basistransformaties, twee schalingen en een krimpdraaiing. Bovendien heeft de fi-



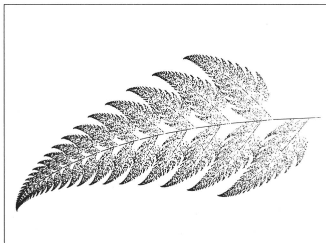
Figuur 5.6 Fractale simulatie van een boomblad

guur een spiegelsymmetrie ten opzichte van een verticale as. Het programma is weer van het bekende type. Het resultaat is afgebeeld in figuur 5.6 en vertoont inderdaad een grote gelijkenis met het biologische origineel.

```
REM ***fractal in de vorm van een boomblad***
REM ***naam:BLAD***
    SCREEN 12 : CLS : RANDOMIZE 11
    WINDOW (-1,-.4)-(1,1.1)
    A=.2 : PHI=1 : KMAX=30000
    R1=.7 : R2=.4 : R3=.7
    Q1=.36 : Q2=1-Q1
    C=COS(PHI) : S=SIN(PHI)
    X=0 : Y=1
    FOR K=1 TO KMAX
        IF INKEY$<>"" THEN END
        R=RND
        SELECT CASE R
            CASE <Q1
                X=R1*X : Y=R1*Y
            CASE <Q2
                X=R2*X : Y=R2*(Y-1)+1
            CASE ELSE
                Z=X : X=R3*(C*X-S*Y)
                IF RND<.5 THEN X=-X
                Y=R3*(S*Z+C*Y)+A
        END SELECT
        PSET (X,Y),10 : PSET (-X,Y),10
    NEXT K : BEEP
END
```

Het meest bekend is het veel afgebeelde varenblad van Barnsley als resultaat van vier lineaire transformaties. Daaronder bevinden zich een centrale vermenigvuldiging en een projectie. De laatste zorgt voor het genereren van de steeltjes. De overige twee zijn affiene transformaties waarbij een vierkantje verkleind en scheef vervormd wordt. Het desbetreffende programma VAREN heeft weer de bekende vorm. Bedenk wel dat met de keuze Q1 = .01 de projectie relatief weinig wordt toegepast ten opzichte van de andere transformaties. Wie hetzelfde programma uitvoert met andere waarden kan voor een verrassing komen te staan, sommige wijzigingen hebben maar weinig invloed op het eindresultaat terwijl kleine veranderingen elders tot grote verstoringen kunnen leiden.

```
REM ***fractal in de vorm van een varenblad***
REM ***data ontleend aan Barnsley's boek***
REM ***naam:VAREN***
    SCREEN 12 : CLS : RANDOMIZE 11
    WINDOW (-1,-4.5)-(11,4.5)
REM ***coefficienten***
```



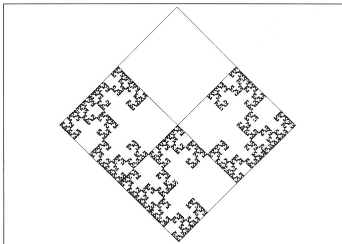
Figuur 5.7 Het varenblad van Barnsley

```
   D1=.16
   A2=.85 : B2=.04 : C2=-.04 : D2=.85 : F2=1.6
   A3=.2 : B3=-.26 : C3=.23 : D3=.22 : F3=1.6
   A4=-.15 : B4=.28 : C4=.26 : D4=.24 : F4=.44
REM ***kansverdeling***
   Q1=.01 : Q2=.86 : Q3=.93
   KMAX=40000 : K=0 : X=0 : Y=0  'start
   DO WHILE K<KMAX AND INKEY$=""
      R=RND
      SELECT CASE R
      CASE <Q1
         X=0 : Y=D1*Y
      CASE Q1 TO Q2
         Z=X : X=A2*X+B2*Y : Y=C2*Z+D2*Y+F2
      CASE Q2 TO Q3
         Z=X : X=A3*X+B3*Y : Y=C3*Z+D3*Y+F3
      CASE >Q3
         Z=X : X=A4*X+B4*Y : Y=C4*Z+D4*Y+F4
      END SELECT
      PSET (Y,X),10
      K=K+1
   LOOP : BEEP
END
```

5.4 Een familie van fractals in een vierkant

Het laatste programma van dit hoofdstuk, SQUAREF, maakt het mogelijk een uitgebreide serie fractals te maken. Het is iets uitvoeriger dan de meeste programma's, maar de wiskundige kern is weer van het bekende type. We beginnen met een vierkant waarvan de hoekpunten de volgende coördinaten hebben: A(2,0), B(-2,0), C(0,-2) en D(0,2). Het centrum van het vierkant is de oorsprong (0,0). Voor het chaosspel gebruiken we drie gelijkvormigheidstransformaties. De eerste T_1 voert het grote vierkant over in het kleinere vierkant waarvan A en O de overstaande hoekpunten zijn. Dit kan op acht manieren worden gerealiseerd, die we kunnen indiceren met de getallen 0, 1, 2, ..., 7. Het type 0 is een gewone centrale vermenigvuldiging ten opzichte van A met de factor 1/2. Daarbij gaat B dus over in het hoekpunt A. Bij het type 1 ondergaat het grote vierkant eerst een draaiing over 90° voordat de centrale vermenigvuldiging ten opzichte van A wordt toegepast. Bij type 2 draaien we het grote vierkant eerst over 180° (of we voeren een puntspiegeling ten opzichte van O uit) en bij type 3 draaien we over 90° in tegengestelde zin. De typen 4, 5, 6 en 7 stemmen, afgezien van een lijnspiegeling, met de vorige vier overeen. In totaal zijn er dus acht mogelijkheden. De transformatie T_2 komt overeen met T_1, maar de rol van A wordt overgenomen door B. De transformatie T_3 ten slotte voert het grote vierkant over in het kleinere, onderste vierkant met C en O als overstaande hoekpunten. Ook bij T_2 en bij T_3 zijn er acht mogelijkheden, totaal dus 8 * 8 * 8 = 512 verschillende mogelijkheden. Aan het begin van het programma kan een willekeurige mogelijkheid vanaf het toetsenbord gedefinieerd worden. De kern van het programma is weer de hoofdlus die uit een beperkt aantal regels bestaat. Uiteraard moeten er bij de aanvang van het programma nogal wat gegevens ter beschikking zijn, zoals een lijst van coëfficiënten voor elk van de acht transformatietypen. Het programma bevat bovendien nog een paar verfraaiingen die voor zich spreken. Met het programma kunnen in beginsel 512 verschillende fractals gemaakt worden. Daar kunnen wel dubbele of onderling symmetrische figuren tussen zitten maar er blijven genoeg fraaie figuren over. Een van de mogelijkheden is afgebeeld in figuur 5.8.

```
REM ***fractals in een vierkant***
REM ***naam:SQUAREF***
   SCREEN 12 : CLS
   RANDOMIZE 11 : PI=4*ATN(1)
   WINDOW (-3.2,-2.4)-(3.2,2.4)
   PRINT"bij elk drietal getallen uit 0,1,2,...,7 behoort een
                                                   fractal"
   PRINT : INPUT"eerste getal = ",T1
   PRINT : INPUT"tweede getal = ",T2
```



Figuur 5.8 Een fractal van het 'vierkante' type

```
   PRINT : INPUT"derde  getal = ",T3
   T=T1 : L=1 : GOSUB selectie
   T=T2 : L=2 : GOSUB selectie
   T=T3 : L=3 : GOSUB selectie
   DIM E(3),F(3)
   E(1)=1 : F(1)=0 : E(2)=-1
   F(2)=0 : E(3)=0 : F(3)=-1
   X=0 : Y=0 : KMAX=30000
   CLS : GOSUB setframe
   LOCATE 1,1  : PRINT T1  : LOCATE 1,10 : PRINT T2
   LOCATE 1,20 : PRINT T3
REM ***hoofdprogramma***
   FOR K=1 TO KMAX : R=RND
      IF INKEY$<>"" THEN END
      L=1+INT(3*RND)
      Z=X : X=A(L)*X+B(L)*Y+E(L)  : Y=C(L)*Z+D(L)*Y+F(L)
      IF K>16 THEN PSET (X,Y)
   NEXT K : BEEP
END
selectie:
   SELECT CASE T
   CASE 0
      A(L)=1/2 : B(L)=0 : C(L)=0 : D(L)=1/2
   CASE 1
```

```
        A(L)=0 : B(L)=-1/2 : C(L)=1/2 : D(L)=0
    CASE 2
        A(L)=-1/2 : B(L)=0 : C(L)=0 : D(L)=-1/2
    CASE 3
        A(L)=0 : B(L)=1/2 : C(L)=-1/2 : D(L)=0
    CASE 4
        A(L)=-1/2 : B(L)=0 : C(L)=0 : D(L)=1/2
    CASE 5
        A(L)=0 : B(L)=1/2 : C(L)=1/2 : D(L)=0
    CASE 6
        A(L)=1/2 : B(L)=0 : C(L)=0 : D(L)=-1/2
    CASE 7
        A(L)=0 : B(L)=-1/2 : C(L)=-1/2 : D(L)=0
    END SELECT
RETURN
setframe:
    LINE (-2,0)-(0,2),4 : LINE -(2,0),4
    LINE -(0,-2),4 : LINE -(-2,0),4
    LINE (-1,-1)-(1,1),4 : LINE (1,-1)-(-1,1),4
RETURN
```

6 Dynamische systemen

6.1 Inleiding

In dit hoofdstuk wordt een aantal programma's behandeld die afkomstig zijn uit wetenschapsgebieden die momenteel sterk in de belangstelling staan. Ze vallen onder de gemeenschappelijke noemer van *dynamische systemen* en zijn in de regel wiskundige modellen van verschijnselen die als *orde en chaos* aangeduid worden. De achterliggende theorie is buitengewoon boeiend, er zijn talloze boeken en artikelen aan gewijd, van goed leesbaar populair-wetenschappelijk tot moeilijke kost voor specialisten. Bij een dynamisch systeem kunnen we allereerst denken aan een bewegend mechanisch systeem, de slinger van een oud uurwerk, of aan de beweging van de hemellichamen. Een dynamisch systeem kunnen we modelleren als een punt dat volgens een vaste wet in een vlak of in de ruimte beweegt. Een aardig beeld is dat van een transportbedrijf of een autobusdienst waarbij het voertuig, zeg de autobus, volgens een bepaalde wetmatigheid een bepaalde route beschrijft die slechts van het gekozen beginpunt P_0 afhangt. Daarbij letten we alleen op de stopplaatsen $P_1, P_2, P_3, \ldots$ Het geheel van stopplaatsen noemen we een *baan*, de baan van P_0. Een baan kan eeuwig doorgaan, en een baan bestaat dus uit oneindig veel baanpunten. Het probleem is doorgaans om op basis van de gegeven wetmatigheid langs theoretische of grafische weg een overzicht te vormen van de verschillende typen banen.

Er zijn namelijk nogal wat mogelijkheden. Een baan kan zich na een paar stappen sluiten. Dit leidt tot een periodieke baan met slechts een beperkt aantal stopplaatsen. Een baan kan tot een limietpunt naderen, en een baan kan in het oneindige verdwijnen. Ten slotte kunnen de baanpunten op een schijnbaar chaotische manier heen en weer blijven

springen, een chaotische baan. Wat daaruit resulteert kan een puntenwolk zijn die voor een deel chaotisch is, maar voor een deel ook een herkenbare structuur kan hebben.

In de praktijk hebben we vaak te maken met evenwichtssituaties, banen die slechts uit een enkel punt bestaan. Zulke evenwichten kunnen net als in de mechanica stabiel of instabiel zijn, een knikker in een putje of een dubbeltje op zijn kant. Een instabiel evenwicht betekent dat een kleine verandering van het beginpunt leidt tot een baan die zich steeds meer van het beginpunt verwijdert. Stabiliteit betekent dat een kleine storing in het begin een baan geeft die in de buurt van het evenwicht blijft en er ten slotte weer terugkeert. Hetzelfde geldt ook voor de zogenaamde periodieke evenwichten, banen die uit een beperkt aantal punten bestaan. Dezelfde stopplaatsen worden telkens opnieuw bezocht.

De voorbeelden die hier worden behandeld, zijn hoofdzakelijk van het autobustype. De autobus is een punt op het beeldscherm, een baan is een lange rij van punten op het scherm, vaak vele duizenden. Om de banen op een overzichtelijke wijze te kunnen beoordelen kan op doeltreffende wijze van kleuren gebruik gemaakt worden. Op deze wijze ontstaan grafische beelden die enerzijds een wetenschappelijke betekenis kunnen hebben, maar die door hun soms verrassende grafische kwaliteiten ook opgevat kunnen worden als voorbeelden van computerkunst.

De wetmatigheid waaraan de banen onderworpen worden, is van dezelfde soort als in het vorige hoofdstuk. Een baanpunt P wordt beschreven als een beeldpunt met de coördinaten (x,y). Het coördinatenstelsel is daarbij niet van belang, dat kan door ons op een willekeurige wijze gekozen worden. Bij $P(x,y)$ behoort een volgend punt $P'(x',y')$, waarbij x' en y' berekend kunnen worden volgens formules van het type:

(6.1) $x' = f(x,y) \qquad y' = g(x,y)$

Hierbij zijn f en g gegeven wiskundige functies.

Een baan kan dan op de volgende wijze geconstrueerd worden. We kiezen een willekeurig beginpunt. De coördinaten van dat punt substitueren we in de rechterleden van (6.1) en we voeren de berekening van x' en y' uit. Daarmee zijn dan de coördinaten van het eerstvolgende baanpunt bepaald. Voor het volgende baanpunt herhalen we deze berekening terwijl we nu in de rechterleden van (6.1) de zojuist gevonden coördinaten x' en y' invullen, dus $x = x'$ en $y = y'$. Zo gaan we door met het herhalen van dezelfde elementaire berekening.

6.2 Populatiedynamica

Tegenwoordig staat het milieu sterk in de belangstelling, en terecht. De oude methoden om insekten te bestrijden met chemicaliën falen meer en meer omdat de insekten resistent worden en de chemische bestrijdingsmiddelen schadelijk zijn voor mens en dier, voor wie ze niet bedoeld zijn. Om betere methoden toe te kunnen passen moeten we meer weten over de natuur, over de voortplantingsmechanismen van insekten, enzovoort. De populatiedynamica, een actueel onderdeel van de biologie, houdt zich juist bezig met dergelijke problemen. Biologen bedienen zich daarbij van wiskundige modellen, zoals het model van Verhulst, een wiskundige beschrijving van geremde groei. Aangezien er veel publikaties over zijn verschenen, gaan we er hier niet verder op in. Wel wordt een verwant model behandeld dat niet alleen biologisch relevant is, maar ook als computermodel erg interessant is.

We gaan uit van een populatie, zeg insekten, die zich seizoensgewijs voortplant. Het relatieve aantal noemen we x_n waarbij n het volgnummer is van de generatie. Het aantal van de volgende generatie is daaraan rechtevenredig, zodat we $x_{n+1} = c\, x_n$ kunnen schrijven. De factor c is niet constant, maar hangt af van de reeds aanwezige aantallen van de huidige en de vorige generaties. Naarmate er meer individuen zijn, is de natuurlijke remming door het milieu des te groter, bijvoorbeeld omdat meer voedsel wordt opgegeten. Dat betekent dat a kleiner wordt naarmate x_n en x_{n-1} groter worden. In het hier gegeven voorbeeld GROEIM1 kiezen we $c = a\,(1-x^2_{n-1})$, waarbij a een constante is. Het complete model is dus:

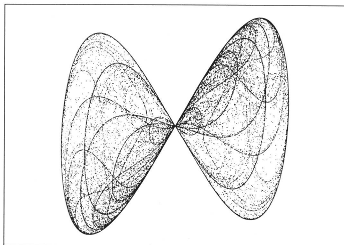
(6.2) $$x_{n+1} = a\, x_n\, (1 - x^2_{n-1})$$

Op het eerste gezicht lijkt dat een heel ander model dan (6.1), maar wanneer we naast x_n een tweede variabele invoeren zoals $y_n = x_{n+1}$, blijkt dat het toch om hetzelfde modeltype gaat. We kunnen namelijk (6.2) herschrijven als

(6.3) $$\begin{aligned} x_{n+1} &= y_n \\ y_{n+1} &= a\, y_n\, (1 - x_n^2) \end{aligned}$$

We vergeten de biologie nu maar en gaan verder met het model (6.3) zonder acht te slaan op de biologische betekenis. Zo kunnen we x en y onbekommerd negatief laten zijn. Wel beperken we ons gemakshalve bij de keuze van a tot het interval (0,2). Door het systeem (6.3) is door elk beginpunt (x_0,y_0) een baan bepaald die we in het x,y-vlak kunnen afbeelden als een stippellijn. In het programma GROEIM1 kiezen we $a = 1.95$, en als beginpunt, vrij willekeurig, $x = y = .1$. In dit geval hebben

we te maken met een chaotische baan waarbij de baanpunten zich kriskras over het beeldscherm bewegen. Naarmate echter meer punten zijn afgebeeld, komt er enige structuur in het plaatje en ontstaat een zogenaamde *vreemde aantrekker* (in het Engels: strange attractor). Het mechanisme is enigszins vergelijkbaar met dat van het in het vorige hoofdstuk besproken chaosspel. Daar werd een baan gevormd uit een toevallige combinatie van twee of meer transformaties van het type (6.1), hier is de vreemde aantrekker het resultaat van een baan die behoort bij een enkele transformatie. De baan die het resultaat is van 400.000 punten, wordt weergegeven in figuur 6.1.



Figuur 6.1 Een vreemde aantrekker bij een model van geremde groei

```
REM ***een model van geremde groei***
REM ***naam:GROEIM1***
REM ***x''= a x' (1-x*x) ***
    SCREEN 12 : CLS
    WINDOW (-2,-1.5)-(2,1.5)
    A=1.95 : KMAX=400000
    X=.1 : Y=.1
    FOR K=1 TO KMAX
        IF INKEY$<>"" THEN END
        Z=X : X=Y : Y=A*Y*(1-Z*Z)
        L=POINT (X,Y)
        IF L<15 THEN L=L+1 ELSE L=1
```

```
        PSET (X,Y),L
    NEXT K : BEEP
END
```

In het programma is ter bevordering van het visuele effect gebruik gemaakt van de techniek van de verlopende kleuren, zoals is beschreven bij de programma's FRACMC6 en FRACMC7. Wie daaraan geen behoefte heeft, kan in het voorgaande programma de 11de en 12de regel schrappen en in de 13de regel het kleurattribuut verwijderen of bijvoorbeeld $l = 14$ (geel) kiezen.

Het programma GROEIM2 is een meer gebruikersvriendelijke versie van het vorige programma. Eerst wordt een vierkant grafisch venster gedefinieerd met een blauwe achtergrondkleur en een helwitte voorgrondkleur. Het model (6.3) wordt onderzocht voor een reeks van 16 waarden van de parameter a, oplopend van $a = 1.35$ tot aan 1.99. De begeleidende tekst geeft onder meer aan welke waarde van a in onderzoek is. Gekozen is een vast beginpunt $x = y = .01$. We zien dat voor $a < 1.5$ de baan eindigt in een limietpunt. Uit de wiskundige theorie volgt dan $x = y = \sqrt{(1\text{-}1/a)}$.

Overschrijdt a de waarde 1.5, dan gaat de baan over in een eivormige gesloten kromme, hetgeen wijst op een soort periodiek gedrag. Die limietfiguur kunnen we ook opvatten als een aantrekker. Het lijkt namelijk of een baan die ergens begint door de limietfiguur aangetrokken wordt. Een limietpunt of limietlus wordt daarom wel een gewone aantrekker genoemd. Als a echter een zekere waarde overschrijdt, kan de gewone aantrekker als het ware ontaarden in een vreemde aantrekker, een figuur opgebouwd uit lussen in lussen in lussen in ..., een fractale meetkundige figuur.

```
REM ***een model van geremde groei***
REM ***naam:GROEIM2***
REM ***x''= a x' (1-x*x) ***
    SCREEN 12 : CLS
    GOSUB text
    FOR K=1 TO 16 : A=1.35+K*.04
        VIEW (200,40)-(600,440),1,15
        WINDOW (-1.7,-1.7)-(1.7,1.7)
        LOCATE 3,1 : PRINT "A =";
        PRINT USING "||.||";A
        X=.01 : Y=.01  'start
        FOR N=1 TO 100+K*K*100
            Z=X : X=Y : Y=A*Y*(1-Z*Z)
            PSET (X,Y)
        NEXT N
        BEEP : GOSUB text : A$=INPUT$(1) : CLS
    NEXT K : SCREEN 0
```

```
END
text:
    LOCATE 22,1 : PRINT"druk op een toets"
    LOCATE 23,1 : PRINT"voor vervolg"
RETURN
```

6.3 Conservatieve systemen

In de mechanica kent men het begrip van een conservatief dynamisch systeem. Conserveren betekent behouden en dat slaat meestal op de in het systeem aanwezige energie. Stellen we ons een mathematische slinger voor als een zwaar punt opgehangen aan een stijf massaloos koord en verwaarlozen we de luchtweerstand, dan wordt bij de slingerbeweging telkens potentiële en kinetische energie uitgewisseld. De som van de twee energievormen blijft daarbij voortdurend constant, een behoudswet, zodat met recht van een conservatief systeem gesproken kan worden.

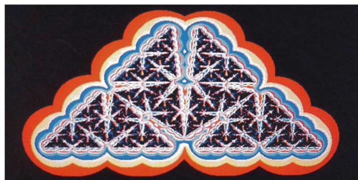
Natuurlijk is dat een idealisatie want in de praktijk zijn er allerlei dempende factoren, zoals de luchtweerstand en de reactie van het ophangtouw of ophangpunt. Dat betekent een voortdurend weglekken van energie, de zogenaamde *dissipatie* van energie. Bij de beweging van de hemellichamen is de energiedissipatie zo gering dat die nagenoeg verwaarloosd kan worden. Om de beweging van de hemellichamen te begrijpen bedienen de geleerden zich van talloze wiskundige modellen. Omdat niet alle modellen geschikt waren voor theoretisch onderzoek, heeft men vaak van de computer gebruik moeten maken. Een van de eerste en meest succesvolle computerexperimenten op dat gebied werd uitgevoerd door de Franse astronoom Michel Hénon. Het door hem opgestelde model heeft de vorm (6.1) en luidt:

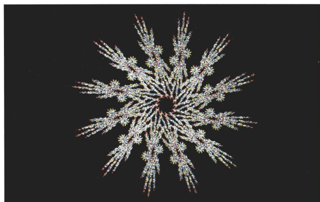
$$(6.4) \qquad \begin{aligned} x_{n+1} &= a\,x_n - b\,(y_n - x_n^2) \\ y_{n+1} &= b\,x_n + a\,(y_n - x_n^2) \end{aligned}$$

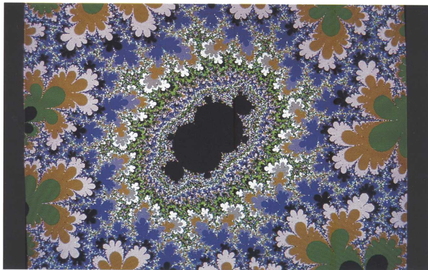
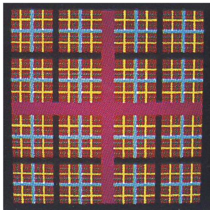
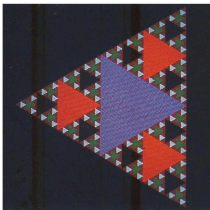
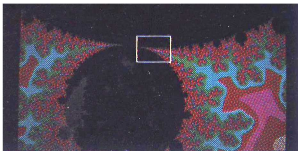
De betekenis van de coëfficiënten a en b is $a = \cos(\alpha)$ en $b = \sin(\alpha)$, waarbij α een gegeven hoek is.

Wanneer x_n en y_n erg klein zijn, kunnen de kwadraten in het rechterlid verwaarloosd worden en blijft er een lineaire transformatie over die we kennen als een draaiing om de oorsprong over de hoek α. Dat betekent dat een baan die vlakbij de oorsprong begint, praktisch gesproken een cirkelbaan is die voortdurend rondloopt. Past de hoek α een geheel aantal malen op de volle omtrek, 2π dus, dan is de baan periodiek en bestaat hij uit een beperkt aantal punten die steeds opnieuw doorlopen



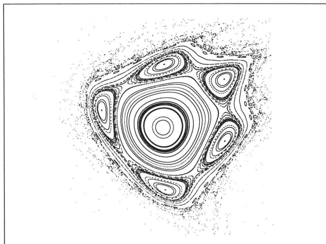




worden. Is algemener α/π een rationaal getal, dan is de baan periodiek. Is de verhouding echter onmeetbaar, dan wordt de cirkelbaan dicht met baanpunten overdekt. De invloed van de kwadratische termen in (6.4) is verrassend. Het model (6.4) is conservatief in de zin dat een kleine cirkel door de transformatie wordt veranderd in een kleine ellips met dezelfde oppervlakte. Wat in de fysische werkelijkheid behoud van energie is, komt in een mathematisch model als (6.4) overeen met het behoud van oppervlakken. Hénon heeft er doelbewust naar gestreefd het meest eenvoudige conservatieve model van het type (6.1) te ontwerpen waarbij naast de lineaire alleen kwadratische termen toegestaan zijn.

Het blijkt dat in zijn model een grote rijkdom aan banen is. Er zijn talloze periodieke evenwichtsposities, er zijn banen die zich oplossen in een beperkt aantal punten en er zijn banen van een elliptische vorm. Het programma HENON1 biedt daarvan een globaal overzicht. Telkens genereert een willekeurig gekozen beginpunt een baan waarvan 1000 punten getoond worden in een bepaalde kleur. Sommige banen dreigen naar het oneindige weg te vluchten, maar het programma heeft daarvoor een ingebouwde veiligheid. Er bestaan ook zogenaamde chaotische banen, banen die op een onvoorspelbare wijze overal tussendoor lijken te krioelen. Het loont de moeite het programma met verschillende waarden van de hoek α of met verschillende waarden van a uit te voeren. In het



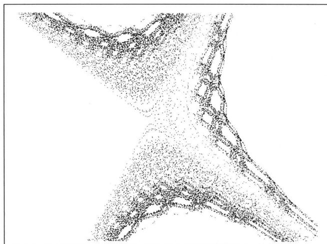
Figuur 6.2 Banen in het model van Hénon

programma is $a = .24$ gekozen, maar andere waarden leveren eveneens interessante plaatjes op. Een resultaat wordt getoond in figuur 6.2.

```
REM ***Henon's kwadratisch systeem***
REM ***een aantal toevallig gekozen banen***
REM ***naam:HENON1***
    SCREEN 12 : CLS : RANDOMIZE 11
    WINDOW (-1.6,-1.2)-(1.6,1.2)
    A=.24 : B=SQR(1-A*A) : M=0
    DO WHILE INKEY$="" AND M<100
        X=-1+2*RND : Y=-1+2*RND : COL=1+INT(15*RND)
        FOR N=1 TO 1000
            IF ABS(X)<1 AND ABS(Y)<1 THEN PSET (X,Y),COL
            Z=X : X=X*A-(Y-X*X)*B
            Y=Z*B+(Y-Z*Z)*A
            IF ABS(X)+ABS(Y)>10 THEN EXIT FOR
        NEXT N
        M=M+1
    LOOP : BEEP
END
```

Het is fascinerend om te zien hoe op het beeldscherm van het model (6.3) een enkele chaotische baan gevormd wordt. In het programma HENON2 wordt zo'n baan berekend. We gaan daarbij kijken in de omgeving, een mini-window, van een zogenaamd instabiel periodiek evenwicht. Het is wel nodig enige honderdduizenden baanpunten te berekenen, omdat maar een deel van die punten in het window terechtkomt. Wat we echter zien, lijkt op een fijn kantwerk met opvallende gaten. In de gaten bevinden zich overigens periodieke evenwichten, omgeven door stabiele gesloten banen en ook weer chaos. Figuur 6.3 illustreert het gedeelte van de chaotische baan die past in het gekozen kijkvenster.

```
REM ***een enkele chaotische baan***
REM ***naam:HENON2***
    DEFDBL A,B,H,U-Y : DEFLNG N
    SCREEN 12 : CLS
    A=.24 : B=SQR(1-A*A)  'parameters
REM ***beginpunt en window***
    X0=.575 : Y0=.16 : H1=.08 : H2=.06
    WINDOW (X0-H1,Y0-H2)-(X0+H1,Y0+H2)
REM ***hoofdlus***
    X=X0 : Y=Y0 : N=0
    DO WHILE INKEY$="" AND N<250000
        IF ABS(X-X0)<.8*H1 AND ABS(Y-Y0)<.8*H2 THEN
                                                PSET (X,Y),14
        U=X : V=Y-X*X
        X=A*X-B*V : Y=B*U+A*V
        N=N+1
    LOOP : BEEP
END
```



Figuur 6.3 Een fragment van een chaotische baan

Op een soortgelijke wijze kunnen veel andere dynamische systemen geprogrammeerd worden. In de meeste gevallen zijn de computerresultaten, gewoonlijk in de vorm van plaatjes, alleen interessant voor specialisten. Soms hebben de grafische beelden echter ook een bepaalde artistieke waarde die grenst aan computerkunst.

6.4 Dynamisch systeem of computerkunst

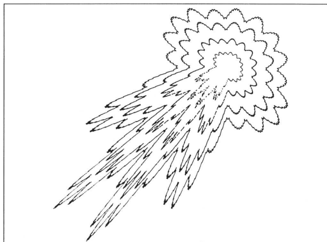
Uit de vele in de literatuur beschreven dynamische systemen kiezen we eerst het volgende model, afkomstig van H.O. Peitgen.

$$x_{n+1} = 2\,x_n - y_n - a\,f(x_n)$$
(6.5)
$$y_{n+1} = x_n$$

Het is een model van een conservatief systeem, ongeacht de betekenis van de functie $f(x)$. Peitgen neemt in het bijzonder de volgende functie:

$f(x) = x$	met $x \le 0$
$f(x) = (\pi/b - 1)\,x$	met $0 < x < b$
$f(x) = b - x$	met $x \ge b$

De waarden van de constanten zijn $a = .37$ en $b = .4$. De betekenis van de op het eerste gezicht ingewikkelde beschrijving van de functie $f(x)$ blijkt wanneer we er een grafiekje van maken, een soort uitgerekte S, bestaande uit drie rechte lijnen. In het desbetreffende programma DYNSYSX1 worden van het systeem (6.5) vier banen getekend die de indruk wekken van een exploderend stuk vuurwerk, zoals in figuur 6.4 wordt weergegeven.



Figuur 6.4 Een artistieke explosie

```
REM ***dynamisch systeem volgens Peitgen, explosie***
REM ***x'=2x-y-af(x), y'=x met***
REM ***f(x)=x voor x<=0 ***
REM ***f(x)=(pi-b)*x/b voor x>0 en x<b ***
REM ***f(x)=pi-b voor x>=b ***
REM ***naam:DYNSYSX1***
   SCREEN 12 : CLS : PI=4*ATN(1)
   DIM X(4),P(4),COL(4)
   WINDOW (-1.4,-1.2)-(.6,.6)
REM ***data***
   A=.37 : B=.4
   X(1)=.05 : P(1)=3000 : X(2)=.1 : P(2)=8000
   X(3)=.16 : P(3)=10000 : X(4)=.21 : P(4)=12000
   COL(1)=9 : COL(2)=10 : COL(3)=12 : COL(4)=14
DEF FNS(X)
   IF X<=0 THEN FNS=X
```

```
   IF X>0 AND X<B THEN FNS=(PI/B-1)*X
   IF X>=B THEN FNS=PI-X
END DEF
REM ***vier banen***
   FOR J=1 TO 4
      X=X(J) : Y=X
      FOR K=0 TO P(J)
         Z=X : X=2*X-Y-A*FNS(X) : Y=Z
         PSET (X,Y),COL(J)
         IF INKEY$<>"" THEN END
      NEXT K
   NEXT J : BEEP
END
```

In het programma DYNSYSX2 gaat het om een enkele chaotische baan volgens het conservatieve systeem:

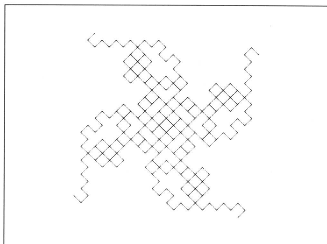
$$x_{n+1} = -y_n + a \sin(x_n)$$

(6.6)

$$y_{n+1} = x_n + a \sin(x_{n+1})$$

waarbij a een willekeurige parameter is. In het programma is a = .08 gekozen en dan blijkt de door het punt (0,3.14) gegenereerde baan chaotisch te zijn. De reden hiervan is onder andere dat het beginpunt vlakbij het instabiele evenwichtspunt $(0,\pi)$ ligt. De chaotische baan is uiterst merkwaardig zoals uit figuur 6.5 blijkt. Ook in dit geval kan men het programma variëren door een andere waarde voor a te kiezen en de baan ergens anders te beginnen. In het programma is als extra gebruik gemaakt van kleuren. Voor een monochrome versie moet alleen de toevoeging na de PSET-opdracht verwijderd worden.

```
REM ***een enkele chaotische baan van***
REM ***x'=-y+f(x), y'=x+f(x') ***
REM ***f(x)=asin x ***
REM ***naam:DYNSYSX2***
   DEFDBL A,X,Y,Z
   SCREEN 12 : CLS
   WINDOW (-64,-48)-(64,48)
REM ***data***
   A=.08 : NMAX=100000
   X=0 : Y=3.14 : N=0
REM ***baan***
   DO WHILE N<NMAX AND INKEY$=""
      N=N+1
      PSET (X,Y),1+INT(N/1000) MOD 15
      Z=X : X=-Y+A*SIN(X)
      Y=Z+A*SIN(Y)
   LOOP : BEEP
END
```



Figuur 6.5 Een zigzaggende chaotische baan

6.5 Het dynamisch systeem van Mira

De wiskundige Christian Mira heeft samen met zijn fysische collega Gumowski veel onderzoek gedaan naar mathematische modellen van het type (6.1). Een van de door hen onderzochte systemen:

(6.7)
$$x_{n+1} = y_n + f(x_n)$$
$$y_{n+1} = - x_n + f(x_{n+1})$$

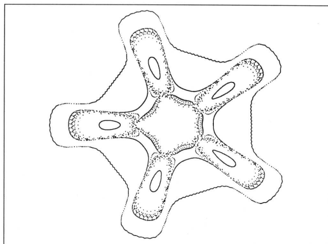
waarbij

$$f(x) = a\,x + 2(1 - a)\,x^2/(1 + x^2)$$

met

$$-1 < a < 1$$

gaf aanleiding tot wonderbaarlijke plaatjes. Een grappige coïncidentie is dat 'mira' in het Latijn 'wonder' betekent. Het systeem (6.7) is weer een voorbeeld van een conservatief dynamisch systeem. De punten (0,0) en (1,0) zijn instabiele evenwichtspunten. In het programma MIRADSX1 worden, vrij willekeurig, drie speciale gevallen voorgesteld. In elk geval wordt een aantal stabiele of chaotische banen afgebeeld. Zoals uit de drie figuren blijkt, kenmerken de stabiele banen zich meestal als gesloten krommen die soms aan eilanden en aan tandwielen doen denken.

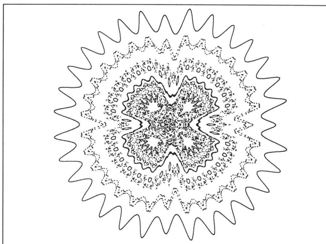


Figuur 6.6 Banen in het Mira-model, a = .3

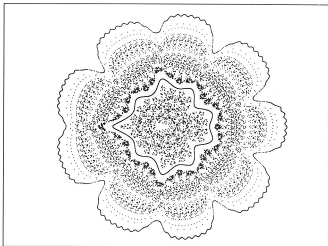
De chaotische banen zijn de gestructureerde wolken die tussen de stabiele banen in liggen.

Het programma is voor de duidelijkheid rechttoe rechtaan geschreven. De kern is de subroutine 'orbit'. Die kern kan men voor elke andere waarde van *a* gebruiken. Afhankelijk van het gekozen beginpunt van een baan moet de WINDOW-opdracht aangepast worden. De drie grafische resultaten van het programma zijn afgebeeld in de figuren 6.6, 6.7 en 6.8.

```
REM ***Mira's dynamisch systeem***
REM ***in een drietal gevallen worden banen afgebeeld***
REM ***naam:MIRADSX1***
   SCREEN 12 : CLS
   WINDOW (-40,-30)-(40,30) : A=.3
   X=7 : Y=0 : P=2 : GOSUB orbit
   X=-12 : Y=0 : P=2 : GOSUB orbit
   X=-21 : Y=0 : P=2 : GOSUB orbit
   BEEP : GOSUB text
   CLS : A=-.05
   X=9.8 : Y=0 : P=1 : GOSUB orbit
   X=20 : Y=0 : P=4 : GOSUB orbit
   X=15 : Y=0 : P=2 : GOSUB orbit
```



Figuur 6.7 Banen in het Mira-model, a = .05



Figuur 6.8 Banen in het Mira-model, a = .18

```
    X=2 : Y=0 : P=4 : GOSUB orbit
    X=18 : Y=0 : P=2 : GOSUB orbit
    X=25 : Y=0 : P=6 : GOSUB orbit
    X=7.5: Y=0 : P=4 : GOSUB orbit
    GOSUB text
    WINDOW (-32,-24)-(32,24)
    CLS : A=.18
    X=8 : Y=0 : P=2 : GOSUB orbit
    X=20 : Y=0 : P=2 : GOSUB orbit
    X=15 : Y=0 : P=6 : GOSUB orbit
    X=5.3 : Y=0 : P=2 : GOSUB orbit
    X=9 : Y=0 : P=2 : GOSUB orbit
    LOCATE 1,1 : PRINT"einde programma"
END
orbit:
    C=2-2*A : KMAX=1000*P
    W=A*X+C*X*X/(1+X*X)
    FOR K=0 TO KMAX
        PSET (X,Y)
        Z=X : X=Y+W : U=X*X
        W=A*X+C*U/(1+U) : Y=W-Z
    NEXT K
RETURN
text:
    LOCATE 1,1 : PRINT"druk op een toets"
    A$=INPUT$(1)
RETURN
```

Zoals opgemerkt is het model (6.7) van Mira conservatief, meetkundig vertaald als oppervlaktebehoudend. Door het wiskundige model met een extra term uit te breiden kan het effect van de *dissipatie* van energie geïmiteerd worden. Fysisch betekent dit dat er energie verloren gaat, bijvoorbeeld door wrijving. Meetkundig wordt dit vertaald in het kleiner worden van de oppervlakken. Bij dergelijke zogenaamde dissipatieve dynamische systemen ziet men vaak het verschijnsel van vreemde aantrekkers, en dat is ook wat men te zien krijgt in het programma MIRADSX2. De iets uitgebreide dissipatieve versie van (6.7) luidt:

$$x_{n+1} = b\,y_n + f(x_n)$$
(6.8)
$$y_{n+1} = -x_n + f(x_{n+1})$$

waarbij

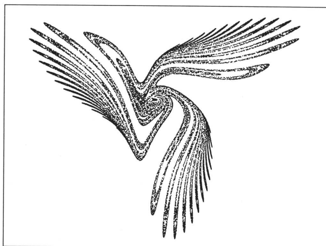
$$f(x) = a\,x + 2(1 - a)\,x^2/(1 + x^2)$$

met

$$-1 < a < 1 \text{ en } b \leq 1$$

Als $b = 1$ is het model conservatief, maar bij $b < 1$ is het model dissipatief. In het genoemde programma geven we van beide gevallen een paar voor-

beelden van chaotische banen. Een enkel fraai resultaat is afgebeeld in figuur 6.9.



Figuur 6.9 Een vreemde aantrekker in het dissipatieve Mira model

```
REM ***chaotische banen in Mira's dynamisch systeem***
REM ***naam:MIRADSX2***
    SCREEN 12 : CLS
    WINDOW (-320,-240)-(319,239)
    DIM COL%(15) : KMAX=50000
    DATA 0,2,10,10,4,4,12,12,12,14,14,14,15,15,15,15
    FOR I=0 TO 15 : READ COL%(I) : NEXT I
REM ***verschillende gevallen***
    A=-.48 : B=.93 : C=2-2*A : D=20
    X=4 : Y=0
GOSUB orbit : GOSUB text
    A=.31 : B=1 : C=2-2*A : D=5
    X=12 : Y=0
GOSUB orbit : GOSUB text
    A=.32 : B=1 : C=2-2*A : D=16
    X=-5 : Y=0
GOSUB orbit : GOSUB text
    A=-.4 : B=.99 : C=2-2*A : D=20
    X=4 : Y=0
GOSUB orbit : GOSUB text
    A=-.4 : B=1 : C=2-2*A : D=14
```

```
   X=4 : Y=0
GOSUB orbit : GOSUB text
   A=-.4 : B=1 : C=2-2*A : D=8
   X=20 : Y=0
GOSUB orbit : GOSUB text
   A=-.2 : B=1 : C=2-2*A : D=14
   X=10 : Y=0
GOSUB orbit
LOCATE 1,1 : PRINT"einde programma"
END
orbit:
   W=A*X+C*X*X/(1+X*X)
   FOR K=0 TO KMAX
      Z=X : X=B*Y+W : U=X*X
      W=A*X+C*U/(1+U) : Y=W-Z
      L=POINT(D*X,D*Y)
      L=1+L MOD 15
      IF K>100 THEN PSET (D*X,D*Y),COL%(L)
      IF INKEY$<>"" THEN EXIT FOR
   NEXT K : BEEP
RETURN
text:
   LOCATE 1,1 : PRINT"druk op een toets voor vervolg"
   A$=INPUT$(1) : CLS
RETURN
```

6.5.1 *Mira met muziek*

Computerprogramma's kunnen op allerlei manieren opgesmukt worden. Een aardige manier is de computer een achtergrondmuziekje ten gehore te laten brengen tijdens de uitvoering van een grafisch programma. In Basic bestaat zelfs de speciale muziektaal PLAY, die het mogelijk maakt een willekeurige melodie in een string te verpakken (zie bijvoorbeeld BQB, paragraaf 10.5). Het is echter ook mogelijk deze opdracht te gebruiken:

```
SOUND freq,time
```

waarmee een enkele toon wordt voortgebracht. Daarin vertegenwoordigt *freq* de frequentie van de toon en *time* de tijdsduur. Beide parameters zijn positieve integers (tussen 0 en 65535). De tijdsduur wordt gewoonlijk gemeten in kloktikken (ongeveer 18 kloktikken per seconde).

In het programma MIRAMUS ziet u hoe met een enkele SOUND-opdracht een soort chaotische melodie wordt gegenereerd tijdens het vormen van een chaotische baan. De truc is om de toonhoogte afhankelijk te laten zijn van de afstand *s* van het baanpunt tot aan de oorsprong. De

daarbij gebruikte formule is slechts een voorbeeld en kan in principe door elke andere formule vervangen worden.

```
REM ***Chaos-muziek gegenereerd door Mira's systeem***
REM ***Naam:MIRAMUS***
   SCREEN 12
   WINDOW (-8,-6)-(8,6)
   A=.3 : B=.999  : C=2-2*A
   Q=2^(1/12) : X=0 : Y=1 : N=0
   LOCATE 1,1 : PRINT"druk op een toets voor einde"
   W=A*X+C*X*X/(1+X*X)
   DO WHILE INKEY$= ""
      IF N>100 THEN PSET (X,Y)
      Z=X : X=B*Y+W : U=X*X
      W=A*X+C*U/(1+U) : Y=W-Z
      S=SQR(X*X+Y*Y) : T=INT(30*S/(S+1))
      SOUND Q^T*416,2
      PSET (X,Y)
      N=N+1
   LOOP
END
```

7 Een spel van leven en dood

7.1 Inleiding

Geleerden en schrijvers van science – fiction – verhalen hebben meermalen met de gedachte gespeeld een abstracte vorm van kunstmatig leven te ontwikkelen. Hierbij stond hun een soort robot voor ogen die in staat is uit onbewerkt materiaal een tweede robot samen te stellen, een exacte kopie van zichzelf. Het denkbeeld van een dergelijk kunstmatig leven (in het Engels: artificial life) is in het midden van deze eeuw nader uitgewerkt toen ook de eerste elektronische rekenmachines gebouwd werden. Bij de theorie van die rekenautomaten werd het abstracte begrip van een zogenaamde *cellulaire automaat* ontwikkeld, een heel algemeen begrip waarover boeken vol geschreven kunnen worden. Beperken we ons echter tot een enkel eenvoudig type, dan kan het aan de hand van een voorbeeld in een paar woorden worden uitgelegd. De 'wereld' bestaat uit de velden van een dambord, de wezens, cellen genaamd, zijn schijven waarvan er 100 zijn, op elk veld één. Een witte schijf betekent een levende cel, een zwarte schijf is een dode cel. Met andere woorden: een veld kan twee *toestanden* hebben, namelijk de toestand 'levend' of de toestand 'dood'. In de wiskundige beschrijving kunnen we die twee toestanden karakteriseren met een bit: 1 voor levend en 0 voor dood.

De evolutie in deze wereld bestaat uit een soort generatiewisseling die plaatsvindt volgens de tikken van een superklok. Bij elke tik veranderen de cellen, allemaal tegelijk, al of niet van aard volgens bepaalde *leefregels*. De eventuele verandering van de toestand van een veld hangt volgens bepaalde regels af van de omgevingssituatie. Een dood veld omringd door een paar levende cellen kan tot leven gewekt worden, anderzijds kan een levende cel bij gebrek aan levende buren te gronde gaan, en ook kan een soort overbevolking tot de dood van een cel leiden.

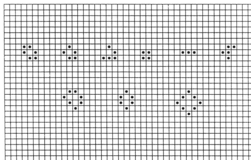
Vijfentwintig jaar geleden ontwierp de Engelse wiskundige Conway een stel eenvoudige regels. Via artikelen van Gardner in de Scientific American werd het 'spel' snel populair, en weldra werd in allerlei universitaire centra menig uur aan Conway's 'Game of Life' opgeofferd. Inmiddels is de aandacht wat verflauwd, maar nu de huiscomputers zoveel krachtiger geworden zijn verdient het spel, ook al om zijn biologische relevantie, hernieuwde aandacht.

7.2 De wetten van Conway

De regels van Conway kunnen als volgt worden beschreven. Een levende cel tellen we als 1, een dode als 0. Een cel wordt omringd door acht buren. Door hun toestanden bij elkaar op te tellen komen we op een getal tussen 0 en 8, dat we s noemen. Om te bepalen of de toestand van een cel voor wijziging in aanmerking komt, kijken we naar de waarde van s:

Als $s < 2$ of $s > 3$, is de cel in de volgende fase dood.
Als $s = 2$, verandert de toestand niet.
Als $s = 3$, is de cel bij de volgende stap levend.

Deze regels betekenen dat voortplanting de samenwerking vereist van drie individuen, anders dan in de traditionele biologie. We zien dat een levende cel die omringd wordt door precies twee of drie levende cellen, ook in de volgende generatie blijft leven. Een levende cel omringd door minstens vier levende cellen, sterft wegens overbevolking. Een levende cel omringd door hoogstens een enkele levende cel, sterft aan eenzaamheid.



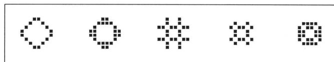
Figuur 7.1 Een aantal objecten in Conway's game of life

Het spel kan al met potlood en ruitjespapier gespeeld worden, en daarbij kan men interessante objecten, groepen van cellen, ontdekken. In figuur 7.1 wordt een aantal voorbeelden weergegeven.

In de bovenste rij zien we van links naar rechts in de oorspronkelijke Engelse benaming 'ship, boat, glider, block, blinker, R-pentomino'. In de onderste rij 'loaf, beehive and traffic light'. De termen behoeven weinig toelichting. Een pentomino (afgeleid van het Griekse woord pente: vijf) is een figuurtje dat bestaat uit vijf cellen op naburige velden.

Sommige blijven van generatie tot generatie onveranderd alsof ze het eeuwige leven hebben, andere veranderen periodiek van vorm. Er zijn echter objecten – zoals de R-pentomino – met een uiterst bizarre evolutie die pas na ruim duizend stappen enigszins overzichtelijk wordt. De 'glider' verandert periodiek van vorm, maar schuift daarbij op diagonale wijze over het rooster. Er zijn ook heel merkwaardige objecten, zoals ruimteschepen, die op hun reis telkens 'gliders' produceren.

In figuur 7.2 wordt de levenscyclus van een periodiek object getoond. In de figuur gaan we uit van het meest linkse object, de volgende generaties zijn telkens een aantal plaatsen naar links geplaatst. Kennelijk zijn er vijf stadia, waarna de oorspronkelijke vierkante ring weer terugkomt.



Figuur 7.2 De levenscyclus van een periodiek object

Eigenlijk kan men alleen maar met een computer een beetje in deze door Conway geschapen wonderwereld doordringen. In een van de boeken van Martin Gardner (Wheels, life and other amusements. Freeman, 1983) kan men daar meer over lezen. Nu beperken we ons tot een paar programma's waarmee verschillende simulaties uitgevoerd kunnen worden. De structuur van het programma is eigenlijk heel eenvoudig. We gaan uit van een vierkant of een rechthoek met *sh* * *sv* posities waarbij *sh* = 120 en *sv* = 90 schermvullend is. De toestanden van de cellen leggen we vast in twee arrays *u*(.,.) en *v*(.,.) waarbij we tijdens een generatiewisseling *u* ongewijzigd laten en in *v* rekenen. In het hoofdprogramma berekenen we aldus voor alle cellen de nieuwe toestand *v*(.,.). In het reset-gedeelte wordt de *v*-array gekopieerd naar de *u*-array en wordt het patroon van de nieuwe generatie op het scherm afgebeeld. De *v*-array is dan weer bruikbaar voor de volgende berekening. Aan het begin van het

programma wordt een aantal eenvoudige administratieve taken uitgevoerd.

```
REM ***Conway's game of life***
REM ***de evolutie van een enkel motief***
REM ***naam:CONWAY1***
    SCREEN 12 : CLS
    DEFINT A-Z
    SH=20 : SV=15 : K=0 : KMAX=16
    WINDOW (3*SH-320,3*SV-240)-(3*SH+319,3*SV+239)
    LINE (-4,-3)-(6*SH+4,6*SV+3),9,B
    DIM U(SH,SV),V(SH,SV)
    REM ***bitpatroon van motief (het verkeerslicht)***
    DATA 0,0,1,0,0
    DATA 0,1,0,1,0
    DATA 1,0,0,0,1
    DATA 0,1,0,1,0
    DATA 0,0,1,0,0
    FOR J=SV/2+2 TO SV/2-2 STEP -1 : FOR I=SH/2-2 TO SH/2+2
        READ U(I,J)
        IF U(I,J)=1 THEN
            LINE (6*I-1,6*J-1)-(6*I+1,6*J+1),10,BF
        END IF
    NEXT I : NEXT J
    LOCATE 1,1 : PRINT"druk op een toets"
    A$=INPUT$(1) : LOCATE 1,1 : PRINT SPACE$(40)
REM ***hoofdprogramma***
    DO WHILE K<KMAX
        FOR I=1 TO SH-1 : FOR J=1 TO SV-1
            S0=U(I,J) : V(I,J)=S0
            S1=U(I-1,J-1)+U(I,J-1)+U(I+1,J-1)
            S2=U(I-1,J)+U(I+1,J)
            S3=U(I-1,J+1)+U(I,J+1)+U(I+1,J+1)
            S=S1+S2+S3
            IF S0=1 THEN
                IF S>3 OR S<2 THEN
                     LINE (6*I-1,6*J-1)-(6*I+1,6*J+1),0,BF
                     V(I,J)=0
                END IF
            END IF
            IF S0=0 AND S=3 THEN
                LINE (6*I-1,6*J-1)-(6*I+1,6*J+1),10,BF
                V(I,J)=1
            END IF
        NEXT J : NEXT I
REM ***reset***
        FOR I=1 TO SH-1 : FOR J=1 TO SV-1
            U(I,J)=V(I,J)
            IF U(I,J)=1 THEN
                LINE (6*I-1,6*J-1)-(6*I+1,6*J+1),10,BF
```

```
            END IF
         NEXT J : NEXT I
         K=K+1 : LOCATE 1,1 : PRINT K : A$=INPUT$(1)
   LOOP
END
```

Het beginpatroon is hier het verkeerslicht, in het programma aanwezig als bitpatroon in een datalijst. Het programma kan gemakkelijk gevarieerd worden. Elk ander beginmotiefje kan als bitpatroon op analoge wijze in het programma geplaatst worden. Let daarbij op de horizontale en verticale grootte van het patroon. Bestaat het beginfiguurtje bijvoorbeeld uit een blokje van 7*7 velden, dan moet de eerste regel waarin het patroon ingelezen wordt, veranderd worden in

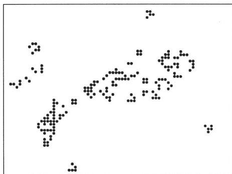
```
FOR J=SV/2+3 TO SV/2-3 STEP -1 : FOR I=SH/2-3 TO SH/2+3
```

Het patroon wordt hierdoor in het centrum van de leefruimte geplaatst, de door SH en SV bepaalde rechthoek. Om eventuele problemen bij het delen te voorkomen moet voor zowel SH als SV een even waarde worden gekozen. Blijft het motief gedurende de evoluties op zijn plaats, dan kunnen we om redenen van snelheid of zuinigheid SH en SV vrij klein kiezen. Gaat het patroon echter als een 'schuiver' aan de wandel, dan is het verstandig zowel SH als SV zo groot mogelijk te nemen. Het programma bevat ook een generatieteller, waardoor het bij elke volgende fase even stopt (A$=INPUT$(1)) waarna het na een willekeurige toetsaanslag vervolgd kan worden.

Let op de evolutie van de R-pentomino, het enige blokje van vijf cellen waarmee iets bijzonders aan de hand is. Pas na meer dan 1000 evolutiestappen ontstaat een overzichtelijke situatie waarbij ondertussen van alles geproduceerd is: blokjes, schuivers, knipperlichten, enzovoort.

Het volledige programma verschilt niet veel van het vorige. In figuur 7.3 ziet u wat er na 160 stappen gebeurd is, een nogal onoverzichtelijke groep cellen waarbij een paar blokjes en schuivers herkenbaar zijn.

```
REM ***Conway's game of life***
REM ***de evolutie van de R pentomino***
REM ***naam:CONWAYR***
   SCREEN 12 : CLS
   DEFINT A-Q,S-Z
   SH=120 : SV=90 : K=0 : KMAX=1200
   WINDOW (2*SH-320,2*SV-240)-(2*SH+319,2*SV+239)
   LINE (-4,-3)-(4*SH+4,4*SV+3),9,B
   DIM U(SH,SV),V(SH,SV)
REM ***beginpatroon***
   DATA 0,1,1
   DATA 1,1,0
   DATA 0,1,0
```

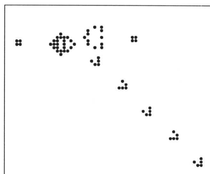


Figuur 7.3 Het nakroost van de R-pentomino na 160 stappen

```
    FOR J=SV/2+1 TO SV/2-1 STEP -1 : FOR I=SH/2-1 TO SH/2+1
        READ U(I,J)
        IF U(I,J)=1 THEN LINE (4*I-1,4*J-1)-(4*I+1,4*J+1),10,BF
    NEXT I : NEXT J
    LOCATE 1,1 : PRINT 0
REM ***hoofdprogramma***
    DO WHILE K<KMAX AND INKEY$=""
        FOR I=1 TO SH-1 : FOR J=1 TO SV-1
            S0=U(I,J) : V(I,J)=S0
            S1=U(I-1,J-1)+U(I,J-1)+U(I+1,J-1)
            S2=U(I-1,J)+U(I+1,J)
            S3=U(I-1,J+1)+U(I,J+1)+U(I+1,J+1)
            S=S1+S2+S3
            IF S0=1 THEN
                IF S>3 OR S<2 THEN
                     LINE (4*I-1,4*J-1)-(4*I+1,4*J+1),0,BF
                     V(I,J)=0
                END IF
            END IF
            IF S0=0 AND S=3 THEN
                LINE (4*I-1,4*J-1)-(4*I+1,4*J+1),10,BF
                V(I,J)=1
            END IF
        NEXT J : NEXT I
REM ***reset***
        FOR I=1 TO SH-1 : FOR J=1 TO SV-1
            U(I,J)=V(I,J)
            IF U(I,J)=1 THEN
                LINE (4*I-1,4*J-1)-(4*I+1,4*J+1),10,BF
            END IF
        NEXT J : NEXT I : K=K+1
```

```
        LOCATE 1,1 : PRINT K
    LOOP
END
```

IJverige puzzelaars, onder andere uit wetenschappelijke kringen, hebben nog veel meer fraais weten op te sporen. In figuur 7.4 wordt de zogenaamde 'glider gun' afgebeeld, een merkwaardig periodiek patroon dat na elke 30 stappen terugkomt en in elke periode tevens een schuiver produceert. Het werd in 1970 gevonden door een team van het Massachusetts Institute of Technology en sierde eens de wiskundige jaarkalender van een internationale uitgever van wiskundeboeken. Met het programma CONWAYS kunnen we het evolutieproces in 150 stappen volgen. De kern van het programma is nog steeds dezelfde, maar het beginpatroon wordt hier op een andere wijze ingelezen en op het scherm geplaatst.



Figuur 7.4 De glider gun na 150 stappen

```
REM ***Conway's game of life***
REM ***naam:CONWAYS***
    SCREEN 12 : CLS
    DEFINT A-Q,S-Z
    SH=60 : SV=50 : K=0 : KMAX=150
    WINDOW (2*SH-320,2*SV-240)-(2*SH+319,2*SV+239)
    LINE (-4,-3)-(4*SH+4,4*SV+3),9,B
    DIM U(SH,SV),V(SH,SV)
REM ***beginpatroon***
    DATA 0,3,0,4,1,3,1,4,11,2,11,3,11,4,12,1,12,5,13,0
    DATA 13,6,14,1,14,5,15,2,15,3,15,4,16,2,16,3,16,4
    DATA 21,4,21,5,21,6,22,3,22,4,22,6,22,7,23,3,23,4
    DATA 23,6,23,7,24,3,24,4,24,6,24,7,25,2,25,3,25,7
```

```
    DATA 25,8,26,5,30,2,30,3,34,4,34,5,35,4,35,5
    FOR I=1 TO 45
        READ X,Y : X=X+3 : Y=Y+36 : U(X,Y)=1
        LINE (4*X-1,4*Y-1)-(4*X+1,4*Y+1),10,BF
    NEXT I
REM ***hoofdprogramma***
    DO WHILE K<=KMAX AND INKEY$=""
        LOCATE 1,1 : PRINT K
        FOR I=1 TO SH-1 : FOR J=1 TO SV-1
            S0=U(I,J) : V(I,J)=S0
            S1=U(I-1,J-1)+U(I,J-1)+U(I+1,J-1)
            S2=U(I-1,J)+U(I+1,J)
            S3=U(I-1,J+1)+U(I,J+1)+U(I+1,J+1)
            S=S1+S2+S3
            IF S0=1 THEN
                IF S>3 OR S<2 THEN
                    LINE (4*I-1,4*J-1)-(4*I+1,4*J+1),0,BF
                    V(I,J)=0
                END IF
            END IF
            IF S0=0 AND S=3 THEN
                LINE (4*I-1,4*J-1)-(4*I+1,4*J+1),10,BF
                V(I,J)=1
            END IF
        NEXT J : NEXT I
REM ***reset***
        FOR I=1 TO SH-1 : FOR J=1 TO SV-1
            U(I,J)=V(I,J)
            IF U(I,J)=1 THEN
                LINE (4*I-1,4*J-1)-(4*I+1,4*J+1),10,BF
            END IF
        NEXT J : NEXT I : K=K+1
    LOOP
END
```

Het is heel interessant om aan het begin de levende cellen op een toevallige wijze te kiezen en vervolgens na te gaan hoe het ontstane patroon evolueert. In veel gevallen ziet u een aantal bekende objecten ontstaan, zoals een blokje van vier, een boot of een brood. Ook knipperlichten, schuivers en nog ingewikkelder figuren kunnen daarbij te voorschijn komen. Een punt van enige zorg is de aanvulling van de regels voor de cellen die zich aan de rand van de rechthoek bevinden. Het is het eenvoudigste om de randcellen altijd als dood te definiëren. Ook kan men overstaande rechthoekszijden identificeren alsof de rechthoek zowel in horizontale zin als in verticale zin omgevouwen en aan elkaar geplakt is. Het programma CONWAY2 verschilt weinig van het programma CONWAY1. Met een andere waarde voor de 'random seed', het getal achter RANDOMIZE, ontstaat een heel ander evolutiebeeld. Natuurlijk kunt u

ook de opdracht RANDOMIZE TIMER gebruiken. Ook de procentuele vulling van de cellen in de eerste fase, in het programma op een waarde van 20 gezet, kan gevarieerd worden.

Op bepaalde computers werkt het programma wellicht wat langzaam. In dat geval kunt u met een kleinere rechthoek (80*60) werken. Overigens maakt een taal als C en ook Assembler een snellere programmatuur mogelijk.

```
REM ***Conway's game of life***
REM ***de evolutie van een toevallige verzameling cellen***
REM ***naam:CONWAY2***
    SCREEN 12 : CLS
    DEFINT A-K,S-Z
    RANDOMIZE 11 : PROC=20
    SH=120 : SV=90
    K=1 : KMAX=1000 : P=PROC/100
    WINDOW (2*SH-320,2*SV-240)-(2*SH+319,2*SV+239)
    LINE (-4,-3)-(4*SH+4,4*SV+3),9,B
    DIM U(SH,SV),V(SH,SV)
    REM ***beginpatroon***
    FOR I=1 TO SH-1 : FOR J=1 TO SV-1
        R=RND
        IF R<P THEN
            U(I,J)=1
            LINE (4*I-1,4*J-1)-(4*I+1,4*J+1),10,BF
        END IF
    NEXT J : NEXT I
    LOCATE 1,1 : PRINT"druk op een toets"
    A$=INPUT$(1) : LOCATE 1,1 : PRINT SPACE$(40)
REM ***hoofdprogramma***
    DO WHILE K<KMAX AND INKEY$=""
    LOCATE 1,1 : PRINT K
    FOR I=1 TO SH-1 : FOR J=1 TO SV-1
            S0=U(I,J) : V(I,J)=S0
            S1=U(I-1,J-1)+U(I,J-1)+U(I+1,J-1)
            S2=U(I-1,J)+U(I+1,J)
            S3=U(I-1,J+1)+U(I,J+1)+U(I+1,J+1)
            S=S1+S2+S3
            IF S0=1 THEN
                IF S>3 OR S<2 THEN
                     LINE (4*I-1,4*J-1)-(4*I+1,4*J+1),0,BF
                     V(I,J)=0
                END IF
            END IF
            IF S0=0 AND S=3 THEN
                LINE (4*I-1,4*J-1)-(4*I+1,4*J+1),10,BF
                V(I,J)=1
            END IF
```

```
        NEXT J : NEXT I
REM ***reset***
        FOR I=1 TO SH-1 : FOR J=1 TO SV-1
            U(I,J)=V(I,J)
            IF U(I,J)=1 THEN
                LINE (4*I-1,4*J-1)-(4*I+1,4*J+1),10,BF
            END IF
        NEXT J : NEXT I : K=K+1
    LOOP
END
```

8 Turtlegraphics

8.1 Inleiding

Veel computergebruikers hebben in hun jonge jaren kennis gemaakt met LOGO, een eenvoudige computertaal bestemd voor kinderen die nog niet toe waren aan het programmeren in hogere programmeertalen zoals Basic en Pascal. In LOGO werd gebruik gemaakt van een zogenaamde schrijvende schildpad (in het Engels: turtle). Deze bewoog zich met een aantal eenvoudige opdrachten over het beeldscherm en kon daarbij een soms heel gecompliceerd plaatje tekenen. Ouder geworden kunnen we LOGO vergeten, maar de tekenende schildpad is nog steeds even nuttig. Weliswaar is het beter om daarbij aan een snel bewegende pen te denken dan aan een in de tuin rondscharrelend diertje. Aangezien 'turtlegraphics' een algemeen bekend begrip is, blijven we toch maar over een 'schildpad' praten.

8.2 De eerste stapjes

Gewapend met de kennis van een hogere programmeertaal kunnen we voor de schildpad een nieuw instructieboekje maken. Dat doen we hier zo eenvoudig mogelijk met de volgende vier regels:

F	Ga een stap h verder en teken.
f	Ga een stap h verder zonder te tekenen.
+	Draai over een gegeven hoek α naar links.
-	Draai over een gegeven hoek α naar rechts.

We nemen aan dat we werken met een x,y-coördinatenstelsel en dat daarin de positie van de schildpad bekend is als x,y,φ.

De hoek φ geeft de richting aan waarin de schildpad gaat bewegen. We stellen ons verder voor dat de schildpad bij elke tik van een superklok een van de instructieregels uitvoert. Een beweging betekent een verplaatsing over een vaste lengte h in de richting waarin de schildpad geplaatst is. Geldt bijvoorbeeld $\varphi = 0$, dan beweegt de schildpad zich in de x-richting over de afstand h. In een Basic-regel is dat eenvoudig $x = x + h$.

Het maken van een tekening bestaat uit het uitvoeren van een reeks van tekenregels, een string opgebouwd uit de symbolen F, f, + en -. De programma's in dit hoofdstuk bestaan in het algemeen uit twee aparte gedeelten. Eerst wordt de string van de af te beelden tekening gedefinieerd, een lange reeks van soms vele duizenden symbolen. Hoe een dergelijke string gemaakt wordt, is een apart verhaal waarop later wordt teruggekomen. We nemen aan dat de string reeds beschikbaar is en de schildpad de daarin opgenomen instructies gaat uitvoeren. Even een eenvoudig voorbeeld. Stel dat de string deze instructies bevat:

```
F+F+F+F+
```

De schildpad dient dan acht instructies uit te voeren. Uiteraard moet de schildpad weten wat de waarde is van h en van α. We kiezen hier $h = 1$ en $\alpha = 90°$. Zetten we de schildpad ergens neer, dan wordt het snel duidelijk dat de schildpad een vierkant met zijde 1 gaat tekenen. De positie van dat vierkant hangt af van de positie waar we de schildpad hebben neergezet. Hierop moet in de diverse programma's altijd gelet worden. Meestal kiezen we $\varphi = 0$ zodat de schildpad in de richting van de positieve x-as begint te schrijven. De x,y-positie aan het begin is in de praktijk vaak een kwestie van experimenteren.

Het draaien over een hoek α kan wiskundig beschreven worden als $\varphi = \varphi \pm \alpha$. Omdat we in de programma's liever met x,y-coördinaten werken, moeten we gebruik maken van de rotatieformules van een vector. De richting φ van de schildpad vertalen we als de vector (p,q), waarbij $p = \cos(\varphi)$ en $q = \sin(\varphi)$. De rotatie van de schildpadvector over de hoek α in de positieve draaizin komt dan neer op:

$$p_1 = p \cos(\alpha) - q \sin(\alpha)$$
$$q_1 = p \sin(\alpha) + q \cos(\alpha)$$

Deze formules zijn vaste bestanddelen van de programma's, waarbij geldt dat $a = \cos(\alpha)$ en $b = \sin(\alpha)$.

Staat de schildpad op de positie (x,y) in de richting φ, dan kan het zetten van een stap h beschreven worden door bij de vector (x,y) de verplaatsingsvector (hp,hq) op te tellen, dus:

$x_1 = x + h\,p$
$y_1 = y + h\,q$

Het programma TURTLE biedt de mogelijkheid met deze regels te oefenen. Screen 12 verschaft onzichtbaar ruitjespapier met horizontaal 640 en verticaal 480 eenheden. De schildpad staat aan het begin in het centrum $x_m = 320$, $y_m = 240$, gericht naar rechts. Het indrukken van de toetsen F en f laat de schildpad bewegen met stapjes h van 10 eenheden, maar in plaats van 10 kan elk ander getal worden gekozen. De toetsen + en - laten we met een kwartdraai corresponderen.

```
REM ***turtlegraphics, stap voor stap***
REM ***naam:TURTLE***
SCREEN 12 : CLS
    WINDOW (-160,-120)-(160,120)
    PRINT"F voorwaarts en tekenen, f voorwaarts niet tekenen"
    PRINT"|+ kwartslag linksom, - kwartslag rechtsom, x einde"
    H=4 : X=0 : Y=0 : P=1 : Q=0 : PSET (X,Y)
    DO
        A$=INPUT$(1)
        SELECT CASE A$
        CASE "F"
            X=X+H*P : Y=Y+H*Q
            LINE -(X,Y)
        CASE "f"
            X=X+H*P : Y=Y+H*Q
            PSET (X,Y)
        CASE "+"
            P1=Q : Q1=-P
            P=P1 : Q=Q1
        CASE "-"
            P1=-Q : Q1=P
            P=P1 : Q=Q1
        END SELECT
    LOOP UNTIL A$="X" OR A$="x"
END
```

De tekenregels van de schildpad kunnen op allerlei manieren gevarieerd worden. Zo kunnen we + interpreteren als een draaiing over een hoek α en - als een draaiing over een andere hoek ß. Ook kunnen we zonder probleem de schildpadtaal uitbreiden met meer symbolen.

8.3 De reisroute

Tot dusver was de aandacht alleen gericht op het lezen en interpreteren van een gegeven symbolenstring. Veel interessanter is het maken van

zo'n string als beschrijving van een meetkundige figuur, een *zelfgelijkvormige fractal* bijvoorbeeld.

Het prototype van een fractal is de bekende lijn van Von Koch, kortweg Kochlijn of Kochkromme, ontdekt aan het begin van deze eeuw. In bijna elk boek over fractals wordt deze kromme uitvoerig beschreven, zodat we er hier kort over kunnen zijn. In het programma TURTLEK wordt die kromme, of liever een benadering daarvan, getekend volgens de methode van turtlegraphics. De meetkundige constructie berust op een reeks van bewerkingen waarbij elke bewerking tot een betere benadering leidt. We beginnen met een horizontaal lijnstuk, de nulde benadering, in coördinaten zich uitstrekkend van (-3,0) tot (0,3). In de volgende stap wordt dit lijnstuk vervangen door een gebroken lijn bestaande uit vier even grote lijnstukjes met de volgende eindpunten: (-3,0), (-1,0), (0, $\sqrt{3}$), (1,0), (3,0). Anders gezegd, het middelste derde gedeelte van de oorspronkelijke lijn wordt verwijderd en vervangen door de opstaande zijden van een gelijkzijdige driehoek. De lijn, de eerste benadering, kan in schildpadtaal worden beschreven door de string:

```
F + F - - F + F
```

waarbij + de betekenis heeft van een draaiing over 60° in tegenwijzerzin en - een overeenkomstige draaiing in wijzerzin. Om de volgende benadering te realiseren passen we op elk van de vier samenstellende lijnstukken dezelfde meetkundige bewerking toe: het middelste derde deel vervangen door de driehoekszijden. Die bewerking betekent in de symboliek van de stringtaal dat we elke letter F moeten vervangen door de gegeven string van acht symbolen:

```
F → F + F - - F + F
```

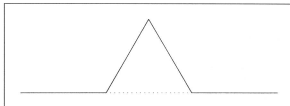
Dit wordt een *produktieregel* genoemd. De string van de tweede benadering, een gebroken lijn van 16 lijnstukjes, bestaat uit 36 symbolen. Zo kunnen we op analoge wijze tot aan de vijfde benadering gaan. We hebben dan een gebroken lijn van 1024 lijnstukjes. Het aantal symbolen in de tekenstring is dan al flink toegenomen. Bij elke volgende benadering hebben we ruwweg viermaal zoveel symbolen nodig. Het exacte aantal per benadering is in tabel 8.1 weergegeven.

Door de beperkte resolutie van het beeldscherm heeft het weinig zin nog verder te gaan. Een andere overweging kan zijn dat er bij de interpretatie van elke volgende string meer rekentijd nodig is. Ook dreigt bij deze methode altijd het gevaar dat er onvoldoende 'stringspace' is. Natuurlijk is daar wel iets aan te doen, maar we vermijden dergelijke foutmeldingen liever door eenvoudig niet zover te gaan.

Benadering	**Aantal symbolen**
0	1
1	8
2	36
3	148
4	596
5	2388

Tabel 8.1

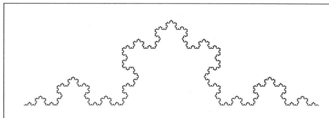
Het spreekt vanzelf dat de staplengte *h* bij het tekenen van een hogere benadering dienovereenkomstig moet worden verkleind. Bij het tekenen van de benaderingen van de Kochlijn moeten we bij elke stap een verkleiningsfactor van 1/3 toepassen om de lijn 'op zijn plaats te houden'. Na deze toelichting zal het programma TURTLEK weinig problemen opleveren. De allereerste figuur, een enkel lijnstuk waaruit alles opgebouwd wordt, draagt in het programma de naam van *axioma*, een term afkomstig uit de wereld van de formele talen, de achterliggende abstracte theorie. De tekenstring, in het programma 'woord' genoemd, wordt stapsgewijs opgebouwd door de string symbool voor symbool te lezen en volgens de IF ... ELSE-constructie de nodige actie te ondernemen.



Figuur 8.1 Het motief van de Koch-lijn

```
REM ***de lijn van Von Koch***
REM ***naam:TURTLEK***
    SCREEN 12 : CLS
    WINDOW (-4,-3)-(4,3)
herhaal:
    CLS : INPUT"ordegetal (keuze uit 0,1,...,5) = ",NMAX
    PI=4*ATN(1) : CLS
REM ***beginpositie en staplengte***
    X=-3 : Y=-.5 : H=6/3^NMAX
```

```
REM ***axioma en productieregel***
    AXIOM$="F"
    PROD$="F+F--F+F"
REM ***vorming van woord***
    WEG$=AXIOM$
    FOR N=1 TO NMAX
        W$=""
        FOR I=1 TO LEN(WEG$)
            S$=MID$(WEG$,I,1)
            IF S$="F" THEN Q$=PROD$ ELSE Q$=S$
            W$=W$+Q$
        NEXT I
        WEG$=W$
    NEXT N
REM ***graphics***
    A=COS(PI/3) : B=SIN(PI/3)
    P=1 : Q=0 : PSET (X,Y)
    FOR I=1 TO LEN(WEG$)
        S$=MID$(WEG$,I,1)
        SELECT CASE S$
        CASE "+"
            P1=A*P-B*Q : Q1=B*P+A*Q
            P=P1 : Q=Q1
        CASE "-"
            P1=A*P+B*Q : Q1=-B*P+A*Q
            P=P1 : Q=Q1
        CASE "F"
            X=X+H*P : Y=Y+H*Q
            LINE -(X,Y)
        END SELECT
    NEXT I
    LOCATE 1,1 : PRINT"toets R voor herhaling programma"
    A$=INPUT$(1)
    IF A$="r" OR A$="R" THEN GOTO herhaal
END
```



Figuur 8.2 Een benadering van de lijn van Von Koch

8.4 Een recursieve constructie van fractals

Met slechts enkele wijzigingen kan in plaats van de Kochlijn een andere fractale kromme getekend worden. Overigens, bedenk wel dat in de praktijk altijd wordt gewerkt met benaderingen van fractals. Een echte fractal is een idealisatie, een soort oneindig ver object dat oneindig veel tekentijd kost. Het is vergelijkbaar met het werken met een onmeetbaar getal zoals π. Een praktische benadering als 3.141593 is ruimschoots voldoende.

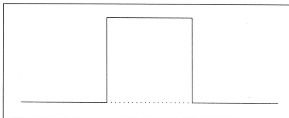
De hoofdzaken van de turtle-methode worden nog even op een rijtje gezet. We beginnen met een *axioma*, de string die past bij de nulde meetkundige benadering. Verder is er een regel die aangeeft hoe een bepaald symbool bij een volgende benadering door een gegeven produktiestring vervangen moet worden, de zogenaamde *produktieregel*. De tekenstring, het woord, dat hoort bij een bepaalde benadering, kan dan gevormd worden. Bij de meetkundige interpretatie van de tekenstring bestaat nog de extra mogelijkheid aan een of meer symbolen een andere betekenis te geven. Dit kan soms leiden tot geheel andere meetkundige figuren.

Een heel eenvoudige variant van de Kochkromme ontstaat door het middelste derde gedeelte van de basislijn te vervangen door drie zijden van een vierkant, zoals afgebeeld in figuur 8.3. De overeenkomstige produktieregel luidt:

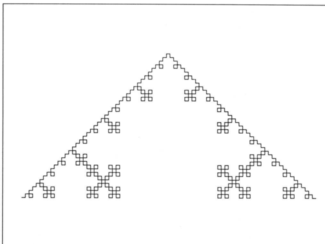
F → F + F - F - F + F

waarbij de + en - nu de betekenis hebben van een kwartdraai. Het desbetreffende programma is TURTLEK1 en een benadering van de vierde orde ziet u in figuur 8.4

```
REM ***kwadratische variant van de Kochlijn***
REM ***naam:TURTLEK1***
   SCREEN 12 : CLS
   WINDOW (-.6,-.3)-(.6,.6)
   NMAX=4 : PI=4*ATN(1)
REM ***beginpositie en staplengte***
   X=-.5 : Y=0 : PHI=0 : H=.0125
REM ***axioma en productieregel***
   AXIOM$="F"
   PROD$="F+F-F-F+F"
REM ***vorming van woord***
   WEG$=AXIOM$
   FOR N=1 TO NMAX
      W$=""
      FOR I=1 TO LEN(WEG$)
         S$=MID$(WEG$,I,1)
```



Figuur 8.3 Het motief van de kwadratische Kochlijn

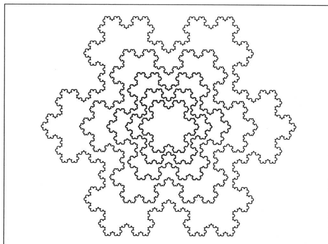


Figuur 8.4 Een kwadratische variant van de Kochlijn

```
            IF S$="F" THEN Q$=PROD$ ELSE Q$=S$
            W$=W$+Q$
        NEXT I
        WEG$=W$
    NEXT N
REM ***graphics***
    PSET (X,Y)
    FOR I=1 TO LEN(WEG$)
        S$=MID$(WEG$,I,1)
        SELECT CASE S$
```

```
        CASE "+"
           PHI=PHI+PI/2
        CASE "-"
           PHI=PHI-PI/2
        CASE "F"
           X=X+H*COS(PHI)  :  Y=Y+H*SIN(PHI)
           LINE -(X,Y)
        END SELECT
    NEXT I
END
```

In het volgende programma TURTLEK2 wordt op elke zijde van een regelmatige zeshoek een Kochlijn geplaatst. Dat betekent dat het programma TURTLEK kan worden gebruikt met alleen een iets ander axioma, namelijk F+F+F+F+F+F. Er ontstaat dan een figuur, een Koch-eiland, die de symmetrievorm heeft van een sneeuwvlok. Maar nu doen we nog iets extra's. Dezelfde tekenstring wordt enige malen gebruikt om dezelfde figuur in een verkleinde vorm te tekenen. Het resultaat wordt in figuur 8.5 weergegeven.



Figuur 8.5 De sneeuwvlok van Von Koch

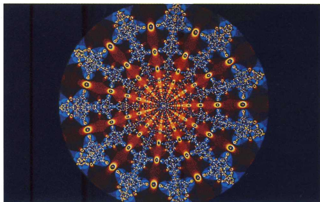
```
REM ***sneeuwvlok, eilanden van Von Koch***
REM ***naam:TURTLEK2***
   SCREEN 12 : CLS
```

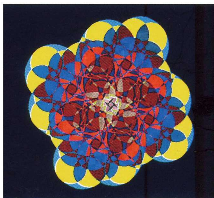
```
    WINDOW (-2.4,-1.8)-(2.4,1.8)
    NMAX=4 : PI=4*ATN(1)
REM ***beginpositie en staplengte***
    H=2/3^NMAX : X=-1 : Y=-SQR(3)
REM ***axioma en productieregel***
    AXIOM$="F+F+F+F+F+F"
    PROD$="F+F--F+F"
REM ***vorming van woord***
    WEG$=AXIOM$
    FOR N=1 TO NMAX
        W$=""
        FOR I=1 TO LEN(WEG$)
            S$=MID$(WEG$,I,1)
            IF S$="F" THEN Q$=PROD$ ELSE Q$=S$
            W$=W$+Q$
        NEXT I
        WEG$=W$
    NEXT N
REM ***graphics***
    A=COS(PI/3) : B=SIN(PI/3)
    FOR J=0 TO 4 : H=.7*H : X=.7*X : Y=.7*Y
        P=1 : Q=0 : PSET (X,Y)
        FOR I=1 TO LEN(WEG$)
            S$=MID$(WEG$,I,1)
            SELECT CASE S$
            CASE "+"
                P1=A*P-B*Q : Q1=B*P+A*Q
                P=P1 : Q=Q1
            CASE "-"
                P1=A*P+B*Q : Q1=-B*P+A*Q
                P=P1 : Q=Q1
            CASE "F"
                X=X+H*P : Y=Y+H*Q
                LINE -(X,Y)
            END SELECT
        NEXT I
    NEXT J
END
```

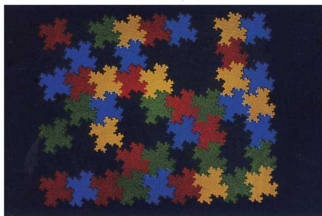
In het bekende boek van Mandelbrot, 'Fractals in Nature', wordt veel aandacht aan de lijn van Von Koch besteed en wordt ook een aantal soortgelijke fractals afgebeeld. Een daarvan is de zogenaamde lijn van Minkowski, een jonggestorven vriend van Einstein en misschien even geniaal. Evenals bij de Kochlijn beginnen we weer met een recht lijntje, maar de produktieregel is nu:

```
F → F-F+F+FF-F-F+F
```

waarbij + en - betrekking hebben op een kwartdraai. Blijkbaar wordt een recht lijntje vervangen door een soort haakse zigzag, een gebroken lijn van acht stukjes zoals geïllustreerd in figuur 8.6.

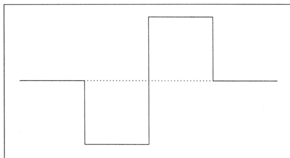




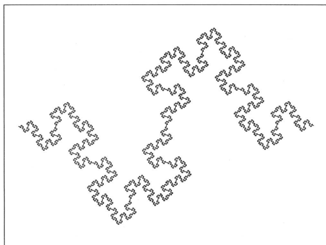




Bij de stapsgewijze vergroting van de tekenstring neemt het aantal elementen toe volgens het rijtje 1, 14, 118, 950, 7606, 60854, ... Tellend vanaf 0 komen we niet verder dan de vierde orde, omdat de vijfde stap doorgaans een foutmelding 'out of string space' zal geven. De vierde orde geeft echter al een zeer bevredigend plaatje, nog nauwkeuriger heeft geen praktische zin. Het resultaat van het desbetreffende programma, TURTLEM, is afgebeeld in figuur 8.7.



Figuur 8.6 Het motief van de lijn van Minkowski



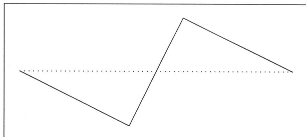
Figuur 8.7 De lijn van Minkowski

```
REM ***de lijn van Minkowski***
REM ***naam:TURTLEM***
   SCREEN 12 : CLS
   WINDOW (-4,-3)-(4,3) : PI=4*ATN(1)
   PRINT"kies de orde als 1,2,3 of 4"
   INPUT"orde = ",NMAX : CLS
   IF NMAX=0 OR NMAX>4 THEN BEEP : END
REM ***beginpositie en staplengte***
   X=-3 : Y=0 : H=6/4^NMAX
REM ***axioma en productieregel***
   AXIOM$="F"
   PROD$="F-F+F+FF-F-F+F"
REM ***vorming van woord***
   WEG$=AXIOM$
   FOR N=1 TO NMAX
      W$=""
      FOR I=1 TO LEN(WEG$)
         S$=MID$(WEG$,I,1)
         IF S$="F" THEN Q$=PROD$ ELSE Q$=S$
         W$=W$+Q$
      NEXT I
      WEG$=W$
   NEXT N
REM ***graphics***
   P=1 : Q=0
   PSET (X,Y)
   FOR I=1 TO LEN(WEG$)
      S$=MID$(WEG$,I,1)
      SELECT CASE S$
      CASE "+"
         P1=-Q : Q1=P
         P=P1 : Q=Q1
      CASE "-"
         P1=Q : Q1=-P
         P=P1 : Q=Q1
      CASE "F"
         X=X+H*P : Y=Y+H*Q
         LINE -(X,Y)
      END SELECT
   NEXT I
END
```

8.5 Een tegelwand van fractale tegels

In het programma TURTLEK3 wordt een lijnstuk telkens vervangen door een gebroken lijn bestaande uit drie gelijke stukken in een onderling loodrechte positie (zie figuur 8.8). Onthoud dat de zigzag symmetrisch is ten opzichte van het oorspronkelijke lijnstuk. De schildpad heeft nu

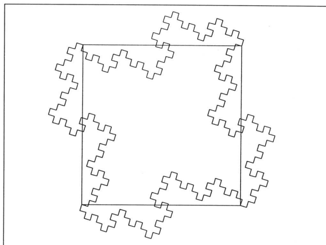
behoefte aan meer symbolen. Aan het begin moet de schildpad gedraaid worden over een hoek α in wijzerzin. Uit de figuur lezen we af dat $\tan(\alpha) = 1/2$, zodat $\sin(\alpha) = 1/\sqrt{5}$ en $\cos(\alpha) = 2/\sqrt{5}$. In de produktieregel geven we de draai over $-\alpha$ aan met het symbool -. Aan het eind van de zigzag moet de schildpad weer in de beginrichting terugkomen, zodat de produktieregel eindigt met +, een draaiing over α in tegenwijzerzin. De kwartdraaien bij *P* en *Q* geven we aan met de extra symbolen L en R (linksom en rechtsom). De volledige produktiestring luidt daarmee -FLFRF+.



Figuur 8.8 Een zigzag van drie gelijke lijnstukjes

Als basisfiguur nemen we een vierkant, FRFRFRF in axiomavorm. Passen we de produktieregel enige malen toe, viermaal bijvoorbeeld, dan wordt het oorspronkelijke vierkant vervangen door een fractale begrenzing overeenkomstig het eiland van Von Koch. In het programma is het oorspronkelijke vierkant zichtbaar gebleven en zijn de stukken die buiten het vierkant steken dankzij de symmetrie van de zigzag identiek aan de binnenste gedeelten. Dat houdt in dat het fractale eiland net als het vierkant waaruit het ontstaan is, gebruikt kan worden als een tegel waarmee het vlak volledig betegeld kan worden.

```
REM ***eiland van Von Koch als een tegel***
REM ***naam:TURTLEK3***
   SCREEN 12 : CLS
   WINDOW (-2.4,-1.8)-(2.4,1.8)
   NMAX=4 : PI=4*ATN(1)
REM ***beginpositie en staplengte***
   A=2/SQR(5) : B=1/SQR(5)
   X=-1 : Y=1 : H=2*B^NMAX
REM ***axioma en productieregel***
   AXIOM$="FRFRFRF"
   PROD$="-FLFRF+"
REM ***vorming van woord***
   WEG$=AXIOM$
```

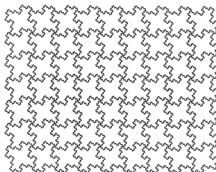


Figuur 8.9 Een fractale tegel

```
    FOR N=1 TO NMAX
        W$=""
        FOR I=1 TO LEN(WEG$)
            S$=MID$(WEG$,I,1)
            IF S$="F" THEN Q$=PROD$ ELSE Q$=S$
            W$=W$+Q$
        NEXT I
        WEG$=W$
    NEXT N
REM ***graphics***
    A=2/SQR(5)  : B=1/SQR(5)
    P=1 : Q=0
    PSET (X,Y),4
    FOR I=1 TO LEN(WEG$)
        S$=MID$(WEG$,I,1)
        SELECT CASE S$
        CASE "L"
            P1=-Q : Q1=P
            P=P1  : Q=Q1
        CASE "R"
            P1=Q  : Q1=-P
            P=P1  : Q=Q1
        CASE "+"
            P1=A*P-B*Q : Q1=B*P+A*Q
            P=P1  : Q=Q1
```

```
        CASE "-"
           P1=A*P+B*Q : Q1=-B*P+A*Q
           P=P1 : Q=Q1
        CASE "F"
           X=X+H*P : Y=Y+H*Q
           LINE -(X,Y),4
        END SELECT
   NEXT I
   PAINT (0,0),4
   LINE (-1,-1)-(1,1),14,B
END
```

In het programma TURTLEK4 wordt de betegeling daadwerkelijk uitgevoerd voor een patroon van 9*7 tegels. Ter verhoging van het visuele effect krijgen de tegels bovendien een kleur, bepaald volgens toeval. Voor elke tegel wordt eerst de omranding in de gekozen kleur getekend en vervolgens wordt de tegel met een PAINT-instructie met dezelfde kleur ingevuld. In figuur 8.10 is een monochrome versie afgebeeld.



Figuur 8.10 Een tegelwand met fractale tegels

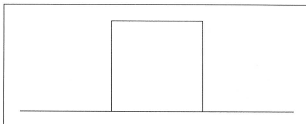
```
REM ***een wand van fractale tegels***
REM ***naam:TURTLEK4***
   SCREEN 12 : CLS
   WINDOW (-22,-20)-(26,16)
   NMAX=4 : RANDOMIZE TIMER
   A=2/SQR(5) : B=1/SQR(5) : H=4*B^NMAX
REM ***axioma en productieregel***
   AXIOM$="FRFRFRF"
   PROD$="-FLFRF+"
REM ***vorming van woord***
   WEG$=AXIOM$
```

```
    FOR N=1 TO NMAX
        W$=""
        FOR I=1 TO LEN(WEG$)
            S$=MID$(WEG$,I,1)
            IF S$="F" THEN Q$=PROD$ ELSE Q$=S$
            W$=W$+Q$
        NEXT I
        WEG$=W$
    NEXT N
FOR IH=-4 TO 4 : FOR IV=-3 TO 3
    COL=1+INT(RND*14)
    X=4*IH : Y=4*IV
    GOSUB graphics
    PAINT (4*IH+1,4*IV-1),COL
NEXT IV : NEXT IH
END
graphics:
    P=1 : Q=0
    PSET (X,Y),COL
    FOR I=1 TO LEN(WEG$)
        S$=MID$(WEG$,I,1)
        SELECT CASE S$
        CASE "L"
            P1=-Q : Q1=P
            P=P1 : Q=Q1
        CASE "R"
            P1=Q : Q1=-P
            P=P1 : Q=Q1
        CASE "+"
            P1=A*P-B*Q : Q1=B*P+A*Q
            P=P1 : Q=Q1
        CASE "-"
            P1=A*P+B*Q : Q1=-B*P+A*Q
            P=P1 : Q=Q1
        CASE "F"
            X=X+H*P : Y=Y+H*Q
            LINE -(X,Y),COL
        END SELECT
    NEXT I
RETURN
```

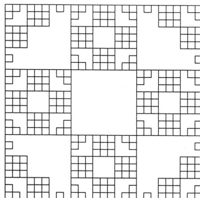
8.6 Van pool tot pool, Sierpinski en Mandelbrot

Figuur 8.11 illustreert een motiefje dat uit zes gelijke lijnstukjes bestaat. Als we daarvoor een produktieregel proberen te bedenken, stuiten we op de moeilijkheid dat het figuurtje niet zomaar als een doorlopende lijn getekend kan worden. We moeten om elk lijnstukje eenmaal te tekenen,

ofwel de pen van het papier halen of een lijnstukje dubbel tekenen. Dat laatste is voor een computer geen bezwaar en we kunnen dan ook de produktieregel FF+F+F+F+FF gebruiken. Met een enkel symbool F als begin groeit de tekenstring snel aan als 1, 11, 81, 571, 4001, 28011, ... zodat we zonder bijzondere maatregelen niet erg ver kunnen komen. In het programma TURTLES is een vierkant als basisfiguur gekozen, en dan gaat het nog harder. Maar in de derde fase krijgen we toch een aardig plaatje (zie figuur 8.12), een soort vierkante versie van de bekende driehoek van Sierpinski.



Figuur 8.11 Een motief voor een Sierpinski-vierkant



Figuur 8.12 Een Sierpinski-vierkant

```
REM ***vierkant van Sierpinski***
REM ***naam:TURTLES***
   SCREEN 12 : CLS
```

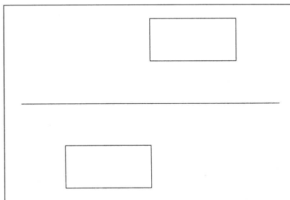
```
   WINDOW (-4,-3)-(4,3)
   NMAX=3 : PI=4*ATN(1)
REM ***beginpositie en staplengte***
   X=-1.5 : Y=-1.5 : H=.12
REM ***axioma en productieregel***
   AXIOM$="F+F+F+F"
   PROD$="FF+F+F+F+FF"
REM ***vorming van woord***
   WEG$=AXIOM$
   FOR N=1 TO NMAX
      W$=""
      FOR I=1 TO LEN(WEG$)
         S$=MID$(WEG$,I,1)
         IF S$="F" THEN Q$=PROD$ ELSE Q$=S$
         W$=W$+Q$
      NEXT I
      WEG$=W$
   NEXT N
REM ***graphics***
   PSET (X,Y) : P=1 : Q=0
   FOR I=1 TO LEN(WEG$)
      S$=MID$(WEG$,I,1)
      SELECT CASE S$
      CASE "+"
         P1=-Q : Q1=P
         P=P1 : Q=Q1
      CASE "-"
         P1=Q : Q1=-P
         P=P1 : Q=Q1
      CASE "F"
         X=X+H*P : Y=Y+H*Q
         LINE -(X,Y)
      END SELECT
   NEXT I : A$=INPUT$(1)
END
```

Het slot van deze serie vormt het programma TURTLEA. Dit maakt een figuur die als Mandelbrot's archipel bekend staat en ook in 'The fractal geometry of nature' is afgebeeld.

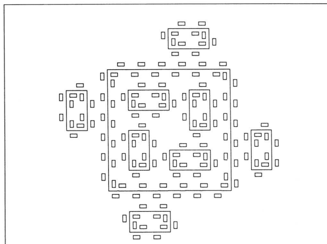
De constructie steunt op de vervanging van een lijnstuk door datzelfde lijnstuk omgeven door twee rechthoekige eilanden, zoals u in figuur 8.13 ziet.

Bij de produktieregel moeten we nu van het symbool 'f' (doe een stap zonder te tekenen) gebruik maken. Met de gebruikelijke betekenis van + en - als kwartdraaien lukt het met de string:

```
F → F-f+FF-F-FF-Ff-FF+f-FF+F+FF+Ff+FFF
```



Figuur 8.13 Het motief van Mandelbrot's archipel



Figuur 8.14 Mandelbrot's archipel

Als uitgangspunt nemen we een vierkant, een axioma dus zoals F+F+F+F. Wie nu denkt klaar te zijn, komt bedrogen uit, zoals men wellicht ontdekt bij een eerste probeersel met potlood en ruitjespapier. Er is

namelijk een tweede produktieregel nodig die aangeeft hoe f bij een volgende fase vervangen moet worden. Dat kan bijvoorbeeld zijn:

```
f → ffffff
```

De programmastructuur verandert hierdoor niet, en er behoeft slechts een extra IF ... THEN-opdracht te worden toegevoegd. Het bezwaar van de beperkte stringruimte doet zich hier duidelijk voelen en verder dan twee stappen komen we met deze methode niet. Het bereikte resultaat, al aardig genoeg, is afgebeeld in figuur 8.14. In 'Computersimulaties' van Lauwerier wordt overigens een methode van backtracking gebruikt waarbij op een zuiniger wijze met het computergeheugen wordt omgegaan.

```
REM ***Mandelbrot's archipel***
REM ***naam:TURTLEA***
    SCREEN 12 : CLS
    WINDOW (-4,-3)-(4,3)
    NMAX=2 : PI=4*ATN(1)
REM ***beginpositie en staplengte***
    X=-1 : Y=-1 : PHI=0 : H=.06
REM ***axioma en productieregel***
    AXIOM$="F+F+F+F"
    PROD1$="F-f+FF-F-FF-Ff-FF+f-FF+F+FF+Ff+FFF"
    PROD2$="ffffff"
REM ***vorming van woord***
    WEG$=AXIOM$
    FOR N=1 TO NMAX
        W$=""
        FOR I=1 TO LEN(WEG$)
            S$=MID$(WEG$,I,1)
            IF S$="F" THEN Q$=PROD1$
            IF S$="f" THEN Q$=PROD2$
            IF S$="+" OR S$="-" THEN Q$=S$
            W$=W$+Q$
        NEXT I
        WEG$=W$
    NEXT N
REM ***graphics***
    PSET (X,Y)
    FOR I=1 TO LEN(WEG$)
        S$=MID$(WEG$,I,1)
        SELECT CASE S$
        CASE "+"
            PHI=PHI+PI/2
        CASE "-"
            PHI=PHI-PI/2
        CASE "F"
            X=X+H*COS(PHI) : Y=Y+H*SIN(PHI)
```

```
        LINE -(X,Y)
      CASE "f"
        X=X+H*COS(PHI) : Y=Y+H*SIN(PHI)
          PSET (X,Y),0
      END SELECT
   NEXT I
END
```

9 Julia sets

9.1 Inleiding

De zogenaamde Julia sets (set is een wiskundige vakterm voor verzameling) vormen een familie van fractals die allemaal volgens hetzelfde formalisme geconstrueerd zijn. Ze zijn voor het eerst beschreven door de Franse wiskundige Gaston Julia, die er omstreeks 1919 met een lang artikel een wetenschappelijke prijs mee won. Julia had nauwelijks een idee hoe de door hem bestudeerde objecten eruitzagen. Fractals waren in die tijd weinig bekend, veel meer dan de kromme van Von Koch was er niet. Pas een vijftigtal jaren later, toen de rekenmachines ter beschikking kwamen, konden de Julia fractals op kleurige wijze op het beeldscherm zichtbaar gemaakt worden. Nu, weer een paar decennia later, kan men ze in alle soorten en maten in luttele minuten op de huiskamercomputer vertonen, als een verslag van een reis in wiskundig sprookjesland. Aan het onderwerp zijn veel boeken gewijd en programmatuur, vaak in de vorm van shareware, is er in overvloed. In dit hoofdstuk bespreken we een paar eenvoudige programma's waarmee dergelijke sets gemaakt kunnen worden.

9.2 De complexe rekenwijze

Om de wiskundige structuur van de Julia sets enigszins te kunnen begrijpen is kennis nodig van complex rekenen, een soort rekenwijze van punten of vectoren (x,y) in een x,y-coördinatenstelsel. Wie daarmee een beetje vertrouwd is, wordt beloond omdat allerlei rekenwerk, onmisbaar in de programma's, plotseling logisch en begrijpelijk wordt. Overigens

kan men de programma's natuurlijk ook zonder kennis van de complexe rekenwijze gebruiken, maar het blijft dan wel een beetje toveren. Om degenen die deze rekenwijze niet kennen tegemoet te komen, leggen we in het kort de basisprincipes uit. Alles berust in zekere zin op een zogenaamde *imaginaire eenheid* die traditioneel als i wordt geschreven. Daarbij geldt de afspraak:

$$i^2 = -1$$

Dat betekent dat we i kunnen opvatten als de vierkantswortel uit -1, een begrip waarvoor in de wereld van de gewone getallen geen plaats is. De verzameling van de gewone getallen breiden we nu uit door er niet alleen i aan toe te voegen, maar ook alles wat gevormd wordt door de getallen, inclusief i, bij elkaar op te tellen, te vermenigvuldigen en te delen. De resultaten hiervan zijn zogenaamde *complexe getallen* die we schrijven als $x + yi$ of $x + iy$. Met complexe getallen rekenen we op de gewone wijze die we op school geleerd hebben, maar zodra we ergens i^2 tegenkomen, schrijven we er meteen -1 voor. Een voorbeeld:

$$(3 + 2i)^2 = 9 + 12i + 4i^2 = 5 + 12i$$

en

$$5/(2 - i) = 5\ (2 + i)/((2 - i)(2 + i)) = 2 + i$$

In veel theorieën spelen ze een centrale rol, met name in de zogenaamde functietheorie waarvan Julia een der experts was. Van bijzonder belang zijn de zogenaamde analytische functies, functies die onbeperkt differentieerbaar zijn. Men zou kunnen stellen dat de complexe getallen uitgevonden zijn om de worteltrekking uit negatieve getallen mogelijk te maken. Dat is dus bereikt door de wortel uit -1 als een aparte imaginaire eenheid te introduceren. Het grote wonder is dat het getallensysteem niet verder uitgebreid hoeft te worden, alles blijkt opeens zonder uitzonderingen mogelijk te zijn. Een vierkantsvergelijking heeft altijd twee oplossingen (die mogelijk kunnen samenvallen), en een hogere machtsvergelijking heeft net zoveel wortels als zijn graad bedraagt. Veel bekende elementaire wiskundige functies, veeltermen, goniometrische en logaritmische functies, enzovoort, kunnen uitgebreid worden tot analytische functies $f(z)$ waarbij z een complexe grootheid is. Elke functie bestaat uit een reëel deel en een imaginair deel. Zo kunnen we de functie z^2 splitsen als:

$$z^2 = (x^2 - y^2) + 2\,x\,y\,i$$

en z^3 als:

$$z^3 = (x^3 - 3\,x\,y^2) + (3\,x^2\,y - y^3)\,i$$

We gaan nog even in op de worteltrekking uit een complex getal, omdat we die bewerking nodig hebben voor een van de programma's. We lichten de methode toe aan de hand van een voorbeeld. Om de wortel uit $5 + 12i$ te berekenen stellen we de wortel voor als $x + yi$ en maken we gebruik van de definitie volgens welke het kwadraat van $x + yi$ gelijk moet zijn aan $5 + 12i$, dus:

$$(x + yi)^2 = 5 + 12\ i$$

De gelijkheid van twee complexe getallen betekent dat zowel de reële delen als de imaginaire delen aan elkaar gelijk zijn. De bovenstaande gelijkheid geldt dus voor de volgende twee gelijkheden:

$$x^2 - y^2 = 5$$

en

$$2\ x\ y = 12$$

Uit deze twee vergelijkingen moeten x en y opgelost worden, en dat doen we met een trucje. We maken gebruik van de identiteit:

$$(x^2 + y^2)^2 = (x^2 - y^2)^2 + (2\ x\ y)^2$$

en hieruit volgt:

$$(x^2 + y^2)^2 = 144 + 25 = 169$$

en dus:

$$x^2 + y^2 = 13$$

Van x^2 en y^2 kennen we nu de som en het verschil, zodat x^2 en y^2 snel bepaald kunnen worden als $x^2 = 9$ en $y^2 = 4$. Hieruit volgt $x = \pm 3$ en $y = \pm 2$, maar niet alle mogelijkheden van + en - leveren een oplossing op. We moeten niet vergeten dat $2xy = 12$ ofwel dat het produkt van x en y positief moet zijn. Met deze beperking resteren de twee oplossingen:

$x + y\ i = 3 + 2\ i$ en $\qquad x + y\ i = -3 - 2\ i$

Voor een vierkantswortel uit een complex getal zijn er in het algemeen twee mogelijkheden. Wie de moeite genomen heeft het een en ander na te rekenen, zal later het programma JULIAMC beter begrijpen.

9.3 Complexe getallen als punten

Het is gebruikelijk aan complexe getallen $x + yi$ een meetkundige voorstelling te verbinden van een punt P met de coördinaten x,y in een cartesisch coördinatenstelsel. Bewerkingen met complexe getallen kunnen

dan vertaald worden in meetkundige termen. De afstand van P tot de oorsprong O is gelijk aan $\sqrt{(x^2 + y^2)}$ en die uitdrukking noemen we de *absolute waarde* van het complexe getal z. De notatie is:

$$|z| = \sqrt{(x^2 + y^2)}$$

Dit begrip is de uitbreiding van het bekende begrip van absolute waarden voor gewone getallen. In een iets andere maar in feite equivalente interpretatie zeggen we dat de absolute waarde van z de lengte van de vector OP is. In de complexe notatie kan een cirkel eenvoudig worden beschreven als $|z| = r$, het binnengebied van een cirkel als $|z| < r$, enzovoort. Zijn z_1 en z_2 twee complexe getallen, meetkundig overeenstemmend met de punten P_1 en P_2, dan stemt de absolute waarde van het verschil overeen met de afstand P_1P_2. Dat betekent:

$$|z_1 - z_2| = \sqrt{((x_1 - x_2)^2 + (y_1 - y_2)^2)}$$

Is $F(z)$ een complexe functie, dan komt de overgang van z naar $F(z)$ neer op een overgang van een punt x,y naar een ander punt zoals we dat gewend zijn bij dynamische systemen. Een transformatie van complexe getallen met de vorm:

$$z' = F(z)$$

kan voor reële variabelen x,y geschreven worden als:

$$x' = f(x,y) \qquad y' = g(x,y)$$

en geïnterpreteerd worden als een meetkundige transformatie, een vervorming van vlakke figuren.

Julia bestudeerde het door $z' = F(z)$ bepaalde iteratieve systeem voor het geval $F(z)$ geschreven kon worden als het quotiënt van twee veeltermen:

$$(9.1) \qquad z_{n+1} = F(z_n)$$

Blijkbaar is dat een dynamisch systeem waarbij een beginpunt z of (x,y) een baan genereert van complexe getallen of punten $z_0, z_1, z_2, z_3, \ldots$ die onbeperkt voortgezet kan worden, waarbij z_0 met z overeenstemt. Alles wat we weten van dynamische systemen kunnen we nu gaan toepassen. De baan van z kan periodiek zijn of niet, de baan van z kan stabiel of instabiel zijn, de baan van z kan begrensd zijn of naar het oneindige gaan. Julia lette op die speciale punten z waarvoor de baan op instabiele wijze periodiek was. De verzameling van z, die daardoor gedefinieerd was, zou men later een Julia set noemen. In beginsel definieert elke (analytische) functie $F(z)$ een Julia set, hier kortweg genoteerd als $J(F)$. Julia liet zien dat $J(F)$ in de regel of uit losse puntjes bestond, stofachtig was als los onsamenhangend zand, of dat alle elementen van $J(F)$ on-

derling samenhingen. Verder bewees hij dat $J(F)$ de eigenschappen had van een fractal, gelijkvormig met zichzelf.

9.4 De kwadratische familie

In dit hoofdstuk gaat het voornamelijk om Julia sets die horen bij de kwadratische functie $F(z) = z^2 + c$. Het iteratieproces wordt dus in de complexe rekenwijze beschreven door:

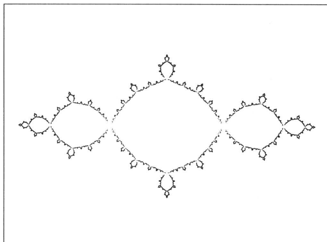
(9.2) $z_{n+1} = z_n^2 + c \qquad c = a + b\,i$

Om de Julia set zichtbaar te maken benutten we in het eerste programma van dit hoofdstuk de eigenschap dat $J(F)$ een afstoter is. Een baan die begint op $J(F)$ blijft erop, maar zodra het beginpunt ergens anders is, gaat de baan ergens anders heen, naar het oneindige of misschien naar een eindig dekpunt. Wanneer het dynamisch systeem geïnverteerd wordt ofwel wanneer we de banen teruglopend volgen, worden de aantrekkers afstoters en omgekeerd. Met andere woorden, een willekeurige baan gaat steeds dichter naar de Julia set toe. De inversie van (9.2) kan in complexe taal gemakkelijk uitgevoerd worden:

(9.3) $z_{n+1} = \pm \sqrt{(z_n - c)}$

Het lijkt even een probleem dat we rechts twee keuzemogelijkheden hebben, maar dat probleem lossen we op de simpelste wijze op door willekeurig een van de twee vierkantswortels te kiezen. De bepaling van de vierkantswortel van een complex getal is hierboven reeds aan de hand van een eenvoudig voorbeeld uitgelegd. Die berekening moet dus bij elke iteratiestap door de computer uitgevoerd worden. In het programma JULIAMC is het een en ander nader uitgewerkt. Het zal duidelijk zijn dat een dergelijk programma zonder enige kennis van de complexe rekenwijze wiskundige hocus pocus bevat. Datzelfde geldt in principe voor alle programma's die zich met Julia sets en Mandelbrot sets bezig houden.

In figuur 9.1 wordt de aldus afgeleide Julia set weergegeven voor de waarden $a = -1$, $b = 0$ na een berekening van 40.000 baanpunten. Het is aantoonbaar dat de Julia set altijd symmetrisch is ten opzichte van de oorsprong, zodat met elk berekend punt ook een ander punt geplot kan worden. Is ook $b = 0$, dan is $J(F)$ ook symmetrisch ten opzichte van de x-as en de y-as. De baanpunten kunnen dan in groepen van vier afgebeeld worden.



Figuur 9.1 De Julia set van z^2-1

```
REM ***Julia set van z*z+c, Monte Carlo methode***
REM ***naam:JULIAMC***
    SCREEN 12 : CLS : RANDOMIZE 11
    WINDOW (-2,-1.5)-(2,1.5)
    A=-1 : B=0  'parameters Julia set
    KMAX = 40000 : K=0 : X=RND : Y=RND
    DO WHILE K<KMAX AND INKEY$=""
        X1=(X-A)/2 : Y1=(Y-B)/2 : R=SQR(X1*X1+Y1*Y1)
        IF RND<.5 THEN
            X=SQR(R+X1) : Y=SQR(R-X1)
            IF Y1<0 THEN Y=-Y
        ELSE
            X=-SQR(R+X1) : Y=-SQR(R-X1)
            IF Y1<0 THEN Y=-Y
        END IF
        IF K>10 THEN PSET (X,Y) : PSET (-X,-Y)
        IF K>10 AND B=0 THEN PSET (X,-Y) : PSET (-X,Y)
        K=K+1
    LOOP
END
```

De Julia sets van de kwadratische familie zijn er in verschillende soorten. $J(F)$ kan uit losse punten bestaan of $J(F)$ hangt samen. In het laatste geval verdeelt $J(F)$ het platte vlak doorgaans in een binnengebied en een buitengebied, maar dat binnengebied kan bij wijze van uitzondering

afwezig zijn. Is dat het geval, dan wordt $J(F)$ een *dendriet* genoemd. Bij een Julia set met een binnengebied kan dat gebied gekleurd worden. Dit leidt tot een zogenaamde *gevulde Julia set*. Het programma JULFILL laat zien hoe de set in dat geval op eenvoudige wijze zichtbaar gemaakt kan worden. Daarbij worden banen van het dynamisch systeem (9.2) gevolgd op de gewone voorwaartse wijze. Uitgeschreven in x,y-coördinaten geldt:

$$
\begin{aligned}
x_{n+1} &= x_n^2 - y_n^2 + a \\
y_{n+1} &= 2\,x_n\,y_n + b
\end{aligned}
\qquad (9.4)
$$

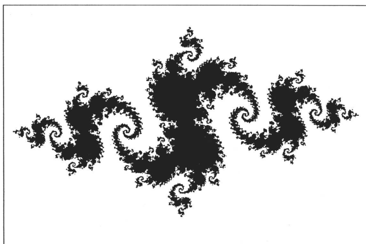
Met de keuze $a = -.8$, $b = .15$ blijkt de Julia set te liggen binnen de rechthoek $-1.6 < x < 1.6$, $-1 < y < 1$. Die rechthoek verschijnt op het beeldscherm als een overeenkomstige rechthoek van pixels, $2n_1$ horizontaal en $2n_2$ verticaal. Heeft n_1 een waarde tussen 50 en 100, dan verschijnt een klein plaatje, maar het kost ook minder tijd. Wie met het onderstaande programma JULFILL experimenteert, kan dus beginnen met een kleine waarde voor n_1, het programma past de waarde van n_2 automatisch aan. Is het resultaat bevredigend, dan kan men tot aan bijvoorbeeld $n_1 = 200$ of $n_1 = 300$ gaan. In principe moet voor elke pixel van de rechthoek op het beeldscherm een test uitgevoerd worden om te controleren of het punt tot het buitengebied behoort of niet. Een punt (x,y) hoort tot het buitengebied als de baan van dat punt naar het oneindige gaat.

In de praktijk hanteren we daarvoor het criterium dat het punt zich na hoogstens 200 stappen (*kmax*) voldoende ver van de oorsprong bevindt, bijvoorbeeld door te eisen dat $x^2 + y^2 > 1000$. Punten die niet aan die eis voldoen, worden geacht tot het binnengebied te behoren. De rekentijd kan wat bekort worden door gebruik te maken van het gegeven dat de Julia set symmetrisch is ten opzichte van de oorsprong. Is ook $b = 0$, dan is $J(F)$ horizontaal en verticaal symmetrisch.

Dat betekent dat we eigenlijk maar de helft of een kwart van alle pixels behoeven te testen. Het programma neemt desondanks veel rekentijd in beslag. Door genoegen te nemen met een kleinere waarde van *kmax*, bijvoorbeeld 100 of zelfs 50, kan men ten koste van een beetje nauwkeurigheid veel tijd winnen. Merk op dat in het programma omwille van de leesbaarheid een GOTO-opdracht opgenomen is. Een fraai resultaat van JULFILL is afgebeeld in figuur 9.2.

```
REM ***gevulde julia set van z*z+c ***
REM ***naam:JULFILL***
    SCREEN 12 : CLS
    WINDOW (-320,-240)-(319,239)
```

```
A=-.8 : B=.15  'parameters
KMAX=200 : DELH=1.6 : DELV=1
N1=250 : N2=INT(N1*DELV/DELH)
LINE (-N1,-N2)-(N1,N2),4,B
IF B=0 THEN N3=0 ELSE N3=-N2
FOR I=0 TO N1 : FOR J=N3 TO N2
    IF INKEY$<>"" THEN END
    X=I*DELH/N1 : Y=J*DELV/N2
    FOR K=1 TO KMAX
        X1=X*X-Y*Y+A : Y1=2*X*Y+B : S=X*X+Y*Y
        IF S>1000 THEN GOTO repeat
        X=X1 : Y=Y1
    NEXT K
    PSET (I,J) : PSET (-I,-J)
    IF B=0 THEN PSET (I,-J) : PSET (-I,J)
repeat:
  NEXT J : NEXT I
END
```



Figuur 9.2 De gevulde Julia set met a = -.8 , b = .15

Bij een aantal Julia sets gaan de banen in het binnengebied naar een limietpunt toe. De rij z_n gaat dan naar een zekere limietwaarde toe. We kunnen die waarde berekenen, maar het is eenvoudiger om gebruik te maken van de constatering dat twee opeenvolgende waarden z_n en z_{n+1} heel weinig verschillen.

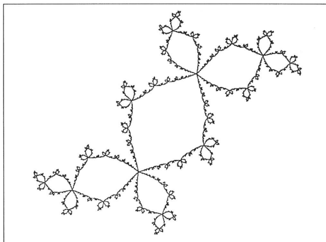
In het programma kunnen we dus testen op $|z_{n+1}-z_n| < \varepsilon$ waarin ε een klein getal als .01 is. Die toets wordt uitgevoerd in het programma JULFILLX, een versie van het vorige programma waarin acht voorbeelden van gevulde Julia sets opgenomen zijn. Bedenk wel dat bij de toets niet de afstandformule zelf wordt gebruikt, maar dat het kwadraat van de afstand wordt genomen. Dat is net zo effectief en het bespaart het berekenen van een vierkantswortel.

```
REM ***gevulde julia set van z:=z^2+c ***
REM ***naam:JULFILLX***
    SCREEN 12 : CLS
    WINDOW (-320,-240)-(319,239)
    PRINT"maak een keuze met een getal uit 1,2,...,8"
    INPUT"getal = ",SEL : CLS
    SELECT CASE SEL
       CASE 1
          A=-1.275 : B=0 : DELH=1.8 : DELV=.8 : N1=240
       CASE 2
          A=-1 : B=0 : DELH=1.7 : DELV=1 : N1=240
       CASE 3
          A=-.75 : B=0 : DELH=1.6 : DELV=1.1 : N1=200
       CASE 4
          A=.25 : B=0 : DELH=1 : DELV=1.3 : N1=120
       CASE 5
          A=-.3905 : B=.5868 : DELH=1.5 : DELV=1.2 : N1=200
       CASE 6
          A=-.1226 : B=.7449 : DELH=1.4 : DELV=1.2 : N1=160
       CASE 7
          A=-.11 : B=.67 : DELH=1.4 : DELV=1.3 : N1=160
       CASE 8
          A=.3 : B=.04 : DELH=1.3 : DELV=1.3 : N1=160
       CASE ELSE
          END
    END SELECT
    N2=INT(N1*DELV/DELH)
    IF B=0 THEN N3=0 ELSE N3=-N2
    FOR I=0 TO N1 : FOR J=N3 TO N2
       IF INKEY$<>"" THEN END
       X=I*DELH/N1 : Y=J*DELV/N2
       FOR K=1 TO 100
          X1=X*X-Y*Y+A : Y1=2*X*Y+B
          S=X*X+Y*Y : S1=(X-X1)*(X-X1)+(Y-Y1)*(Y-Y1)
          IF S>1000 THEN GOTO repeat
          IF S1<.0001 THEN EXIT FOR
          X=X1 : Y=Y1
       NEXT K
       PSET (I,J) : PSET (-I,-J)
       IF B=0 THEN PSET (I,-J) : PSET (-I,J)
repeat:
    NEXT J : NEXT I
END
```

9.5 Op het scherp van de snede

Een Julia set $J(F)$ is als een ragdunne lijn, nauwelijks te treffen door een willekeurig geplaatst punt. Punten dichtbij $J(F)$ genereren banen die van $J(F)$ weglopen. Het is alleen daarom al moeilijk een goed plaatje van een Julia set te maken. Figuur 9.1 laat een op het eerste gezicht redelijke illustratie zien, maar in de wetenschap dat de Julia set in dit geval een continue fractale lijn is, is het resultaat onbevredigend. We zien hier en daar onderbrekingen, gaten die in het toevalsproces van JULIAMC niet gevuld worden, althans niet binnen een redelijke tijd. Er is dus behoefte aan meer verfijnde methoden. Een van die methoden is de zogenaamde *boundary scan*. We stellen ons de Julia set, een set met een binnengebied en een buitengebied, voor als een nog onzichtbare puntenverzameling op het beeldscherm. Over het scherm bewegen we een klein rechthoekje ABCD met horizontale en verticale zijden. Wanneer het binnengebied van de rechthoek geen punten van $J(F)$ bevat, zullen de vier hoekpunten tegelijkertijd ofwel in het binnengebied van $J(F)$ of in het buitengebied van $J(F)$ liggen. In alle andere gevallen bedekt de rechthoek een klein stukje van $J(F)$ en kunnen we veronderstellen dat het centrum van ABCD vlakbij de Julia set ligt. Voor de nauwkeurigheid dient de rechthoek heel klein te zijn, zo klein als mogelijk is binnen de beperkingen van de beschikbare rekentijd. In het programma JULBSC is deze techniek nader uitgewerkt. De kern van het programma is gelijk aan die van JULFILL. Het programma is uitgevoerd voor een Julia set die in de literatuur bekend staat als *het konijn van Douady*, een van de pioniers op het gebied van de theorie van de Julia sets. Het konijn, of de cactus, is afgebeeld in figuur 9.3.

```
REM ***Julia set van z^2+c, boundary scanning***
REM ***naam:JULBSC***
   SCREEN 12 : CLS
   WINDOW (-320,-240)-(319,239)
   A=-.12 : B=.74
   DELH=1.3 : DELV=1.3
   KMAX=80 : N1=200 : N2=INT(N1*DELV/DELH)
   IF B=0 THEN N3=0 ELSE N3=N2
   LINE (-N1,-N2)-(N1,N2),9,B
   FOR I=0 TO N1 : FOR J=-N3 TO N2
      T=0 : IF INKEY$<>"" THEN END
      X=(I+.5)*DELH/N1 : Y=(J-.5)*DELV/N2
      GOSUB jcycle
      X=(I+.5)*DELH/N1 : Y=(J+.5)*DELV/N2
      GOSUB jcycle
      X=(I-.5)*DELH/N1 : Y=(J+.5)*DELV/N2
      GOSUB jcycle
      X=(I-.5)*DELH/N1 : Y=(J-.5)*DELV/N2
```

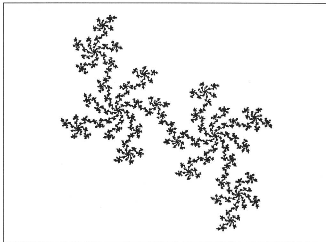


Figuur 9.3 Het konijn van Douady

```
      GOSUB jcycle
      IF T=0 OR T=4 THEN GOTO repeat
      PSET (I,J),14 : PSET (-I,-J),14
      IF B=0 THEN PSET (I,-J),14 : PSET (-I,J),14
repeat:
   NEXT J : NEXT I
END
jcycle:
   FOR K=1 TO KMAX
      X1=X*X-Y*Y+A : Y1=2*X*Y+B
      IF X1*X1+Y1*Y1>4 THEN
         T=T+1 : EXIT FOR
      END IF
      X=X1 : Y=Y1
   NEXT K
RETURN
```

Er bestaat een wiskundige formule die een schatting geeft van de afstand van een willekeurig punt in het buitengebied van een Julia set tot de Julia set. Voor de afleiding van die formule verwijzen we naar ' Een wereld van fractals' . Hier volstaan we met de vermelding van het desbetreffende programma JULDIST. De formule met de getalwaarde *dist* wordt gebruikt met een factor c die het beste proefondervindelijk kan worden bepaald. In het programma is c = .2 gekozen, hetgeen een dunne Julia

lijn oplevert. Door een wat grotere waarde te gebruiken kan de Julia set wat 'opgedikt' worden. Het programma blijkt bijzonder effectief te zijn bij een stofachtige Julia set waar andere methoden soms falen. Het resultaat voor $a = .111$ en $b = .666$ is afgebeeld in figuur 9.4.



Figuur 9.4 De Julia set met $a = .111$ en $b = .666$

```
REM ***Julia set met de afstandsformule***
REM ***naam:JULDIST***
    DEFDBL A-D,S-Y
    SCREEN 12 : CLS
    WINDOW (-320,-240)-(319,239)
    A=.111 : B=.666   'parameters julia set
    C=.2  'coefficient afstandsfunctie
    DELH=1.4 : DELV=1.4
    N1=200 : N2=INT(N1*DELV/DELH)
    LINE (-N1,-N2)-(N1,N2),1,B
REM ***hoofdprogramma***
    FOR I=0 TO N1 : FOR J=-N2 TO N2
        IF I=0 AND J=0 THEN GOTO repeat
        IF INKEY$<>"" THEN END
        X=I*DELH/N1 : Y=J*DELV/N2
        U=1 : V=0
        FOR K=0 TO 200
            X1=X*X-Y*Y+A : Y1=2*X*Y+B
            U1=2*(U*X-V*Y) : V1=2*(U*Y+V*X)
```

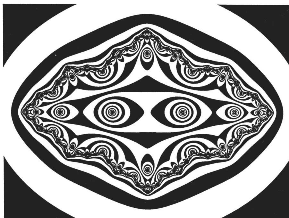
```
S1=X1*X1+Y1*Y1+1E-10 : S2=LOG(S1)
S3=SQR(U1*U1+V1*V1+1E-10)
IF S1>256 OR S3>256 THEN
   DIST=SQR(S1)*S2/S3
   IF DIST<C*DELH/N1 THEN
      PSET (I,J),14 : PSET (-I,-J),14
   END IF
   GOTO repeat
END IF
   X=X1 : Y=Y1 : U=U1 : V=V1
        NEXT K
repeat:
   NEXT J : NEXT I
END
```

9.6 Julia sets in kleurige banden

We hebben gezien dat de baan van een punt *P* in het buitengebied van een kwadratische Julia set *J*(*F*) naar het oneindige gaat. Ligt *P* dicht tegen *J*(*F*) aan, dan duurt het aanvankelijk even voordat de baanpunten weglopen. De mate waarin dat gebeurt, kunnen we meten met een *vluchtgetal* (in het Engels: escape number). Is *R* een groot getal, zodanig dat het buitengebied van de cirkel $|z| = R$ zeker tot het buitengebied van de Julia set behoort, dan kan het vluchtgetal gedefinieerd worden als het eerste rangnummer waarvoor het *n*de baanpunt z_n buiten de genoemde cirkel terechtkomt, dus waarvoor geldt $|z_n| > R$. Het is al goed wanneer we $R = 2$ nemen, maar er ontstaan mooiere plaatjes wanneer *R* groter gekozen wordt, 10 of nog wat hoger.

In het standaardprogramma JULIAP gebruiken we het dynamisch systeem (9.4) zoals in JULFILL. Als illustratief voorbeeld is $a = -.55$, $b = 0$ gesteld. In dat geval is de Julia set *J* een beetje ellipsachtige kronkellijn. Binnen een rechthoek waarin de Julia set gelegen is, toetsen we pixel voor pixel hoe de daardoor gegenereerde baan zich gedraagt. Is de pixel een punt in het buitengebied van *J*(*F*), dan kunnen we het vluchtgetal *k* omzetten in een kleurwaarde, bijvoorbeeld als $1 + k$ mod 15. Het aantal baanpunten beperken we daarbij tot maximaal 100, in het programma aangeduid met *kmax*. Banen die beginnen in het binnengebied blijken begrensd te zijn, en althans in dit speciale geval te eindigen in een enkel limietpunt, een puntaantrekker. De positie van dat punt zou wiskundig berekend kunnen worden, maar het is gemakkelijker om te eisen dat de afstand tussen twee opeenvolgende baanpunten kleiner is dan een gegeven klein getal. We eisen derhalve dat $|z_k - z_{k-1}| < \varepsilon$ waarbij ε bijvoorbeeld .01 is. Het eerste rangnummer *k* waarvoor dit geldt, kunnen we gebruiken om ook voor de binnenbanen een kleur te definiëren. Figuur 9.5

is een monochrome versie van het resultaat. Daartoe kunnen we bijna hetzelfde programma gebruiken. Het is voldoende om alleen die punten te printen waarvoor het vluchtgetal even (of oneven) is. De illustratie is overigens verkregen als laserprint van een virtueel beeldscherm met een resolutie van 1600*1200 pixels.



Figuur 9.5 De kwadratische Julia set met a = -.55, b = 0

In het programma is weer dankbaar gebruik gemaakt van de symmetrie van *J*(*F*). In het gegeven geval hoeft slechts een kwart van de rechthoek aan de toetsing onderworpen te worden. Wie dit standaardprogramma voor andere waarden van *a* en *b* wil uitvoeren, zal daarbij voor de kleuren van het buitengebied geen problemen ondervinden. Het is mogelijk dat de afmetingen van de door *delh* en *delv* bepaalde rechthoek aangepast moeten worden. Om de rekentijd te beperken moet die rechthoek zo klein mogelijk zijn, precies passend om de Julia set. Men kan dat experimenteel bepalen door te beginnen met hele kleine waarden van n_1, 50 bijvoorbeeld. Het plaatje heeft dan de grootte van een flinke postzegel, waaraan in principe alles al te zien is.

```
REM ***Julia set van z*z+c basisprogramma***
REM ***naam:JULIAP***
    SCREEN 12 : CLS
    WINDOW (-320,-240)-(319,239)
    A=-.55 : B=0  'parameters
    DELH=1.6 : DELV=1.2 : KMAX=100
    N1=200 : N2=INT(N1*DELV/DELH)
```

```
    IF B=0 THEN N3=0 ELSE N3=N2
    FOR I=0 TO N1 : FOR J=-N3 TO N2
        IF INKEY$<>"" THEN END
        X=I*DELH/N1 : Y=J*DELV/N2
        FOR K=1 TO KMAX
            X1=X*X-Y*Y+A : Y1=2*X*Y+B
            S=X*X+Y*Y : S1=(X-X1)*(X-X1)+(Y-Y1)*(Y-Y1)
            IF S>1000 THEN L=1+K MOD 15 : GOTO graphics
            IF S1<.0001 THEN L=1+K MOD 15 : GOTO graphics
            X=X1 : Y=Y1
        NEXT K : L=0
graphics:
        PSET (I,J),L : PSET (-I,-J),L
        IF B=0 THEN PSET (I,-J),L : PSET (-I,J),L
    NEXT J : NEXT I
END
```

Op dezelfde wijze kunnen nog veel meer mooie kwadratische fractals op het scherm gezet worden. In het programma JULIAPX is een zestal aardige voorbeelden bij elkaar gebracht.

```
REM ***Julia set van z*z+c, zes voorbeelden***
REM ***name:JULIAPX***
    SCREEN 12 : CLS
    WINDOW (-320,-240)-(319,239)
REM ***kleuren***
    DIM COL(8) : DATA 0,1,9,2,10,4,12,6,14
    FOR I=0 TO 8 : READ COL(I) : NEXT I
    PRINT"maak een keuze met een getal uit 1,2,...,6"
    INPUT"keuze = ",SEL : CLS
    SELECT CASE SEL
        CASE 1
            A=-1.029 : B=.3863 : DELH=2 : DELV=1.2
            KMAX=100 : N1=240 : P=1
        CASE 2
            A=-.75 : B=0 : DELH=1.8 : DELV=1.2
            KMAX=50 : N1=240 : P=1
        CASE 3
            A=-.7454 : B=.1103 : DELH=1.6 : DELV=1
            KMAX=200  : N1=200 : P=5
        CASE 4
            A=-.55 : B=0 : DELH=1.6 : DELV=1.2
            KMAX=100 : N1=200 : P=1
        CASE 5
            A=-.1226 : B=.7449 : DELH=1.7 : DELV=1.5
            KMAX=100  : N1=180 : P=1
        CASE 6
            A=.1 : B=.3 : DELH=1.5 : DELV=1.5 : KMAX=50
            KMAX=100 : N1=160 : P=1
        CASE ELSE
```

```
         END
   END SELECT
   N2=INT(N1*DELV/DELH)
   IF B=0 THEN N3=0 ELSE N3=N2
   FOR I=0 TO N1 : FOR J=-N3 TO N2
      IF INKEY$<>"" THEN END
      X=I*DELH/N1 : Y=J*DELV/N2
      FOR K=1 TO KMAX
         X1=X*X-Y*Y+A : Y1=2*X*Y+B
         S=X*X+Y*Y : S1=(X-X1)*(X-X1)+(Y-Y1)*(Y-Y1)
         IF S>1000 THEN L=1+INT(K/P) MOD 8 : GOTO graphics
         IF S1<.0001 THEN L=1+INT(K/P) MOD 8 : GOTO graphics
         X=X1 : Y=Y1
      NEXT K : L=0
graphics:
      PSET (I,J),COL(L) : PSET (-I,-J),COL(L)
      IF B=0 THEN PSET (I,-J),COL(L) : PSET (-I,J),COL(L)
   NEXT J : NEXT I
END
```

9.7 Julia sets als wandversiering

Het thema van de Julia sets is nagenoeg onuitputtelijk. Bij de kwadratische familie bestaat al een grote verscheidenheid van vormen, maar er zijn in zekere zin net zoveel andere families als er wiskundige functies zijn, oneindig veel dus. Om daaraan mooie plaatjes te kunnen ontlenen is enig wiskundig inzicht nodig. In ' Een wereld van fractals' wordt een aantal fraaie voorbeelden gegeven, maar hier beperken we ons tot een enkel geval, in feite ook weer een rijke familie. Gewapend met het in dit hoofdstuk aangereikte wiskundige gereedschap kunnen we bijvoorbeeld Julia sets van een functie als:

(9.5) $F(z) = z^8 + z^4 + c$

creëren. Eerder is aangegeven hoe z^2 uit z wordt afgeleid. Op precies dezelfde wijze kunnen we z^4 vinden door z^2 te kwadrateren en ten slotte volgt z^8 op analoge wijze uit z^4. Tot zover zijn er geen technische problemen. Net als bij de kwadratische familie heeft een door (9.5) bepaalde fractal een buitengebied dat gekarakteriseerd wordt door banen die naar het oneindige vluchten. In het programma JULIAT kijken we alleen naar het binnengebied en dat gaan we kleuren met een nieuwe techniek. We nemen aan dat de banen in het binnengebied aangetrokken worden door een limietpunt. Dat limietpunt is moeilijk te berekenen en dat doen we dus niet. Wel merken we weer op dat van een binnenbaan twee opeenvolgende punten, zeg z en z' , onderling steeds minder gaan ver-

schillen. We kunnen dus een test ontwerpen die stelt dat |*z-z'* |, meetkundig de afstand tussen de twee baanpunten, ' klein' is. De meest eenvoudige interpretatie daarvan is natuurlijk |*z-z'* |< ε waarbij ε een klein getal is. Nu heeft een zekere auteur met meer fantasie dan wiskundige kennis ooit eens een fout gemaakt door dat ' klein' zijn op een wiskundig onjuiste wijze in het programma te verwerken. Hij werd beloond met onverwacht mooie plaatjes zoals we hier kunnen zien. In het programma schrijven we $z = x + yi$ en $z' = u + vi$. Verder is $d_1 = |x-u|$ en $d_2 = |y-v|$. In plaats van een of ander klein getal ε schrijven we gewoon eps. De eenvoudigste test is dus |*z-z'* | < ε of uitgeschreven in het programma:

```
IF D1*D1+D2*D2<EPS THEN ...
```

Dat werkt overigens goed, maar de plaatjes zijn toch een beetje saai. Een nagenoeg equivalente test is:

```
IF D1<EPS AND D2<EPS THEN ...
```

Zoals opgemerkt, nam iemand abusievelijk:

```
IF D1<EPS OR D2<EPS THEN ...
```

Wiskundig niet zo logisch maar dit leidt wel tot mooiere plaatjes, zij het met een paar kleine defectjes. In het volgende programma is de test nog iets verbeterd om de laatste schoonheidsfoutjes weg te kunnen werken. De test luidt nu:

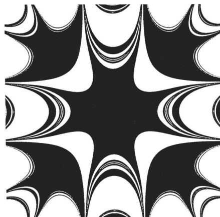
```
IF (D1<P AND D2<Q) OR (D1<Q AND D2<P) THEN ...
```

waarbij *p* een klein getal is (ε in het programma) en *q* een getal dat enige malen groter is (40 ε in het programma).

JULIAT omvat drie voorbeelden die elk een mooi plaatje opleveren. Het loont de moeite zelf te experimenteren met de diverse toetsen. Men kan ε een andere waarde geven, in plaats van 40 een andere factor nemen, enzovoort.

De door (9.4) bepaalde Julia sets hebben een vorm die tussen een vierkant en een regelmatige achthoek in ligt. In sommige gevallen doen ze denken aan tegels. In figuur 9.6 is een enkele tegel afgebeeld. In het programma ziet u hoe daaruit een wandversiering van tegels opgebouwd kan worden.

```
REM ***julia fractal van z:=z^8+z^4+c ***
REM ***name:JULIAT***
   SCREEN 12 : CLS : EPS=.00001
   WINDOW (-320,-240)-(319,239)
   PRINT"maak een keuze uit 1,2,3" : PRINT
```



Figuur 9.6 Een Julia tegel

```
   INPUT""keuzegetal = ",SEL : CLS
   SELECT CASE SEL
   CASE 1
      A=0 : B=.2 : DELH=.95 : DELV=.95
   CASE 2
      A=.3 : B=0 : DELH=.95 : DELV=.95
   CASE 3
      A=-.5 : B=0 : DELH=.95 : DELV=.95
   END SELECT
REM ***kleuren***
   DIM COL(6) : DATA 0,1,9,4,12,2,10
   FOR I=0 TO 6 : READ COL(I) : NEXT I
REM ***grootte plaatje***
   N1=160 : N2=INT(N1*DELV/DELH)
   IF B=0 THEN N3=0 ELSE N3=N2
REM ***hoofdprogramma***
   FOR I=0 TO N1 : FOR J=-N3 TO N2
      IF INKEY$<>"" THEN END
      X=I*DELH/N1 : Y=J*DELV/N2
      FOR K=1 TO 100
         S=X*X+Y*Y
         IF S>40 THEN GOTO herhaal
         X2=X*X-Y*Y : Y2=2*X*Y
         X4=X2*X2-Y2*Y2 : Y4=2*X2*Y2
         X8=X4*X4-Y4*Y4 : Y8=2*X4*Y4
         U=X8+X4+A : V=Y8+Y4+B
         D1=ABS(X-U) : D2=ABS(Y-V)
```

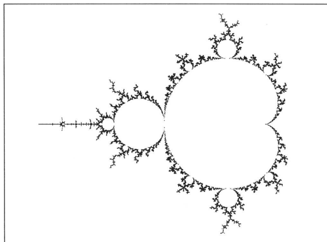
```
         IF D1<EPS AND D2<40*EPS THEN L=1+K MOD 6 : GOTO graphics
         IF D2<EPS AND D1<40*EPS THEN L=1+K MOD 6 : GOTO graphics
         X=U : Y=V
      NEXT K : L=0
graphics:
      PSET (I,-J),COL(L) : PSET (-I,J),COL(L)
      IF B=0 THEN PSET (I,J),COL(L) : PSET (-I,-J),COL(L)
herhaal:
   NEXT J : NEXT I
END
```

10 De Mandelbrot set

10.1 Inleiding

De Mandelbrot set, door Mandelbrot onder de aandacht van het grote publiek gebracht, kan worden beschouwd als een atlas van kwadratische Julia sets. De atlas bestaat uit een enkele kaart, maar van die kaart kunnen details vervaardigd worden met een vergrotingsfactor van miljoenen of nog veel meer. We hebben in het vorige hoofdstuk gezien dat een kwadratische Julia set bepaald wordt door een iteratief dynamisch systeem dat in complexe notatie beschreven kan worden door $z' = z^2 + c$ waarbij c een complex getal is. We schrijven $z = x + yi$ en $c = a + bi$ en tijdens het rekenen houden we altijd de meetkundige interpretatie van de punten (x,y) en (a,b) in een cartesisch coördinatenstelsel in gedachte. Bij elke waarde van c, dus bij elk punt c in het vlak behoort een Julia set die we in dit hoofdstuk als $J(c)$ of kortweg J noteren. Soms is J weinig interessant, soms is J een boeiende wirwar van punten of lijnen. In beginsel kunnen we de Julia sets van de kwadratische familie indelen in twee soorten, de stofachtigen en de lijnachtigen. Zoals de naam al aanduidt, bestaat een stofachtige Julia set uit losse punten en zijn bij een lijnachtige Julia set alle punten onderling verbonden tot een samenhangend geheel.

Al voordat Mandelbrot zich ermee bezig hield kwam men op de gedachte een soort landkaart te maken door in het vlak alle punten c te markeren waarvoor $J(c)$ lijnachtig, en dus samenhangend, was. Dat viel niet mee in een tijd waarin de computers vooral op grafisch gebied nog weinig vermochten. Echter, met een moderne huiscomputer bereikt men binnen enkele minuten een resultaat zoals afgebeeld in figuur 10.1, de zogenaamde Mandelbrot set, kortweg als M aangeduid.



Figuur 10.1 De Mandelbrot set

Vermoedelijk zal de lezer van dit hoofdstuk meer geïnteresseerd zijn in het maken van mooie plaatjes - en daartoe wordt volop gelegenheid gegeven - dan in het begrijpen van de wiskundige achtergrond. Maar toch wordt getracht hier en daar een tipje van de sluier op te lichten zodat men met enige hulp in staat is op een meer gerichte wijze verder te experimenteren in plaats van lukraak iets te proberen. Het is verrassend dat voor sommige toepassingen een klein beetje wiskunde met wat kennis van complex rekenen al voldoende is.

10.2 De Mandelbrot set gevangen in kleuren

Het vervaardigen van een plaatje als in figuur 10.1 steunt op een bepaalde wiskundige eigenschap van $J(c)$. Wanneer de baan van $z = 0$ begrensd blijft, ofwel niet naar het oneindige gaat, behoort het punt c tot M. Net als bij de weergave van een Julia set kunnen we van alle pixels van een gegeven rechthoek systematisch onderzoeken of deze al of niet tot M behoren. Het begin van de baan van 0 kunnen we schrijven als snel aangroeiende veeltermen van c:

$$0 \quad c \quad c^2 + c \quad c^4 + 2c^3 + c^2 + c \quad (c^4 + 2c^3 + c^2 + c)^2 + c$$

We noteren deze als $p_k(c)$ waarbij de telling met $k = 0$ begint. Uiteraard gebruiken we die veeltermen niet als zodanig, bij de berekening passen we telkens de iteratieve betrekking

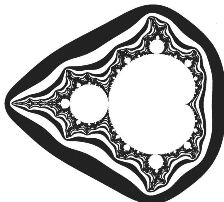
(10.1) $p_{k+1}(c) = p_k(c)^2 + c, \qquad p_1(c) = c$

toe. We zeggen dat de baan van 0 naar oneindig gaat wanneer voor een zekere index m de waarde van $|p_m(c)|$ een vooraf gegeven groot getal r overtreft. We kunnen ook volstaan met $r = 2$, maar om esthetische redenen kan men r beter wat groter nemen, 10 of 100 bijvoorbeeld. De index m wordt het *vluchtgetal* van de baan genoemd. Alle punten c die niet tot M behoren, hebben dus een vluchtgetal. Punten die wel tot M behoren, hebben banen die niet naar oneindig gaan en die dus eeuwig blijven rondzweven. Ze kunnen daarbij soms naar een periodieke aantrekker, een limietcyclus, gaan of ze kunnen een chaotisch gedrag vertonen.

Het vluchtgetal kunnen we gebruiken om het buitengebied van M, alle punten die niet tot M behoren, te kleuren. Evenals bij de Julia set kunnen we uit het vluchtgetal een kleurattribuut afleiden door het bijvoorbeeld MOD 8 te reduceren als men acht kleuren wenst te gebruiken. De Mandelbrot set wordt dan omgeven door kleurige banden. In het programma MANDELX1 zijn nog een paar verkortingen doorgevoerd die een aanzienlijke tijdwinst opleveren. Omdat M symmetrisch is ten opzichte van een horizontale middellijn hoeft slechts de helft van de pixels binnen de omringende rechthoek getoetst te worden. Het meest opvallende gedeelte van M heeft de vorm van een nier of een hart. Bewezen kan worden dat de omtrek daarvan inderdaad een hartlijn of cardioïde is en dat daarbij voldaan is aan:

$$4\,c = 2\,\exp(ti) - \exp(2ti)$$

waarbij t van 0 tot aan 2π loopt. Dit wiskundige gegeven is verwerkt in een eenvoudige toets. Zodra een punt c binnen de cardioïde ligt, hoeft het niet verder getoetst te worden en kijken we naar het volgende punt. Iets soortgelijks geldt voor het cirkelvormige gedeelte links van het hartgedeelte. Dat blijkt een zuivere cirkelschijf te zijn met straal 1/4. Punten binnen die cirkel kunnen dus ook meteen van de onderzoeklijst afgevoerd worden. Voor alle andere punten moeten we de baan nalopen. Daarbij gaan we in het programma niet verder dan 50 punten. Het is niet zinvol verder te gaan wanneer een totaalbeeld van M op het scherm is gewenst.



Figuur 10.2 De Mandelbrot set omringd door banden

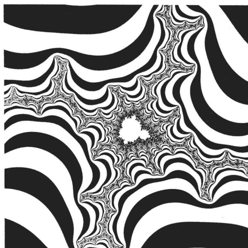
```
REM ***totaalbeeld van de kwadratische Mandelbrot set***
REM ***naam:MANDELX1***
    SCREEN 12 : CLS
    WINDOW (-320,-240)-(319,239)
    DELH=1.6 : DELV=1.34 : AC=-.65
    N1=260 : N2=INT(N1*DELV/DELH)
REM ***kleuren***
    DIM COL(8) : DATA 0,1,9,2,10,4,12,6,14
    FOR I=0 TO 8 : READ COL(I) : NEXT I
REM ***hoofdprogramma***
    LINE (N1+1,N2+1)-(-N1-1,-N2-1),4,B
    FOR I=-N1 TO N1 : A=AC+I*DELH/N1
        FOR J=0 TO N2 : B=J*DELV/N2
            U=4*(A*A+B*B) : V=U-2*A+1/4
            IF U+8*A+15/4<0 THEN L=0 : GOTO repeat
            IF V-SQR(V)+2*A-1/2<0 THEN L=0 : GOTO repeat
            X=A : Y=B : K=0
            DO
                Z=X : X=X*X-Y*Y+A : Y=2*Z*Y+B
                S=X*X+Y*Y : K=K+1
            LOOP UNTIL S>100 OR K=50
            IF K<40 THEN L=1+K MOD 8 ELSE L=0
            IF K>3 THEN
                PSET (I,J),COL(L) : PSET (I,-J),COL(L)
            END IF
repeat:
            IF INKEY$<>"" THEN END
        NEXT J
    NEXT I
END
```

10.3 Details

Bijna hetzelfde programma kan gebruikt worden om een willekeurig detail van M in elke gewenste vergroting op het scherm te zetten. Om tegelijkertijd wiskundig verantwoorde plaatjes te krijgen moet een aantal vuistregels in acht worden genomen. De mooiste details zijn te vinden in de omgeving van de rand van M. Oppervlakkig gezien is M een soort continent omgeven door een archipel van kleine eilandjes die bij nadere beschouwing miniatuurtjes van M blijken te zijn. Dat was tenminste de indruk van Mandelbrot toen hij de figuur voor het eerst te zien kreeg. Nu weten we dat alles met alles samenhangt, verbonden door soms ragdunne draadjes. Bij een verdere vergroting blijken de draden verdikkingen te vertonen die ook weer kleine 'mandelbrotjes' zijn met nog fijnere haren. Het bepalen van een goed plekje is een kunst apart waarop we later terugkomen. Hoe kleiner het detail, des te meer rekentijd benodigd is. Het aantal punten van de te onderzoeken banen moet dan ook opgevoerd worden tot misschien 1000 of meer. Veel hangt af van ervaring. Meestal is het verstandig te beginnen met een klein plaatje ter grootte van een flinke postzegel. Wanneer alles goed gaat en het detail voldoende interessant is om er meer tijd in te steken, kan men het volledige beeldscherm gebruiken en de computer desnoods een nachtje laten doorrekenen. In figuur 10.3 is een dergelijk detail weergegeven. Het programma MANDET biedt de mogelijkheid een willekeurig miniatuurrechthoekje op te blazen tot de grootte van het beeldscherm.

Daartoe moet men aan het programma zowel de coördinaten van het midden van het detail als de halve breedte van de rechthoek opgeven. Het aantal lussen van het hoofdprogramma is gefixeerd op 200, voldoende voor de meeste doeleinden. Alleen om vergrotingen in de orde van miljoenen te maken is een hogere waarde van n nodig of nuttig. De kleuren zijn vastgelegd in een tabel van acht waarden. De gebruiker kan zowel het aantal als de aard van de kleuren wijzigen.

```
REM ***Mandelbrot set, detail***
REM ***naam:MANDET***
    SCREEN 12 : CLS
    DEFDBL A,B,D,S-Z
    WINDOW (-320,-240)-(319,239)
REM ***input of data***
    INPUT"x coordinaat centrum = ",AC
    INPUT"y coordinaat centrum = ",BC
    INPUT"halve breedte = ",DEL : CLS
    N1=200 : N2=INT(.75*N1) : KMAX=200
REM ***kleuren***
    DIM COL(8) : DATA 0,1,9,2,10,4,12,6,14
    FOR I=0 TO 8 : READ COL(I) : NEXT I
```



Figuur 10.3 Een detail met een Mandelbrot baby

```
REM ***hoofdprogramma***
   FOR I=-N1 TO N1 : A=AC+I*DEL/N1
      FOR J=-N2 TO N2 : B=BC+J*.75*DEL/N2
         X=A : Y=B : K=0
         DO
            Z=X : X=X*X-Y*Y+A : Y=2*Z*Y+B
            S=X*X+Y*Y : K=K+1
         LOOP UNTIL S>100 OR K=KMAX
         IF K<40 THEN L=1+K MOD 8 ELSE L=0
         PSET (I,J),COL(L)
         IF INKEY$<>"" THEN END
repeat:
      NEXT J
   NEXT I
END
```

In tabel 10.1 zijn de waarden van een aantal interessante details opgenomen voor experimentele doeleinden.

a	b	delh	kmax
-1.953712	0	.000049	200
-1.927199	0	.0005	200
-1.749057	.000306	.000004	300
-1.28408	.42726	.000625	100
-1.25636	.38032	.008	100
-.91667	.26667	.06	200
-.7789	.1344	.0032	150
-.7777	.1355	.0037	150
-.746371	.098641	.00001	300
-.7392	.1745	.0028	200
-.698	.3785	.004	100
-.160651	1.036793	.00001	600
-.15067	1.04504	.000006	200
-.102324	.95716	.000007	200
-.1011	.9563	.0016	200
-.023262	.999253	.003	100
-.01556	1.02071	.0015	100
.28	.28	.00028	400
.2812	.00948	.00005	300
.2833	.0107	.002	200

Tabel 10.1

10.4 De afstandformule

Het blijkt mogelijk te zijn voor punten in het buitengebied van M de afstand tot de rand van M vrij nauwkeurig te schatten met behulp van een zogenaamde afstandformule. Met een geringe extra moeite kan die afstand uit de door (10.1) bepaalde baan $p_k(c)$ afgeleid worden. De bedoelde schatting is:

(10.2) $d(c,M) = \log(s_1)\ s_2/s_3$

waarbij

$s_1 = |p_k(c)|^2 \quad s_2 = \sqrt{s_1} \quad \text{en} \quad s_3 = |ds_1/dc|$

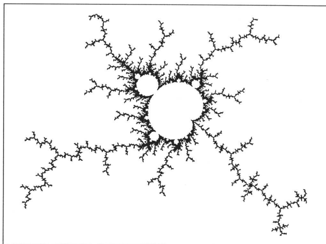
De schatting is bruikbaar wanneer het *k*de baanpunt $p_k(c)$ voldoende ver van M verwijderd is. In het programma MANDIST, waarin het een en ander is uitgewerkt, wordt als eis gesteld dat $s_1 > 128$ is. Om s_3 te kunnen berekenen moeten we de zogenaamde gedifferentieerde baan berekenen, dus van $q_k = dp_k/dc$. Dit wordt in het programma bereikt door tegelijk met elke p_k ook de q_k uit te rekenen volgens de iteratieve formule:

$q_{k+1}(c) = 2\, p_k(c)\, q_k(c) + 1, \quad q_1(c) = 1$

Het programma is vooral geschikt om de fascinerende harige details van M zichtbaar te maken. Die harige tentakels zijn zo dun dat ze meestal ontsnappen aan de door de pixels gedicteerde resolutie van het beeldscherm. Dankzij de afstandformule kunnen ze nu zichtbaar gemaakt worden, zoals figuur 10.4 illustreert. Daartoe moet een test uitgevoerd worden die $d(c,M)$ verbindt aan het oplossend vermogen van het beeldscherm. In het programma schrijven we daarvoor:

```
IF DIST<C*DELH/N1 THEN ...
```

waarbij C een proefondervindelijk te bepalen constante is. Het effect van C is het verdikken van de haren. Voor C = 1 of hoger zijn de haren duidelijk en dik, voor C = .1 zijn ze misschien te dun en ontstaan er gaten. De waarde C = .4 zoals in het programma lijkt een goede middenweg.



Figuur 10.4 Een harig detail

```
IF INKEY$<>"" THEN END
      FOR K=0 TO 100
          X1=X*X-Y*Y+A  :  Y1=2*X*Y+B
          U1=2*(U*X-V*Y)+1  :  V1=2*(U*Y+V*X)
          W1=X1*X1+Y1*Y1
          X=X1  :  Y=Y1  :  U=U1  :  V=V1
```

```
REM ***Detail van Mandelbrot set met afstandsformule***
REM ***naam:MANDIST***
   DEFDBL A-D,U-Z
   DEFINT I-P
   SCREEN 12 : CLS
   WINDOW (-320,-240)-(319,239)
REM ***data of Mset***
   AC=-.16 : BC=1.03  'coordinaten centrum detail
   DELH=.025 : DELV=.75*DELH  'grootte detail
   N1=200 : N2=INT(.75*N1)  'grootte schermbeeld
   CLS : KMAX=200
REM ***main loop***
   FOR I=-N1 TO N1 : FOR J=-N2 TO N2
      A=AC+I*DELH/N1 : B=BC+J*DELV/N2
      X=A : Y=B : U=1 : V=0
      IF INKEY$<>"" THEN END
      FOR K=0 TO KMAX
         X1=X*X-Y*Y+A : Y1=2*X*Y+B
         U1=2*(U*X-V*Y)+1 : V1=2*(U*Y+V*X)
         S1=X1*X1+Y1*Y1+1E-8
         IF S1>128 THEN
            S2=LOG(S1) : S3=SQR(U1*U1+V1*V1)
            DIST=SQR(S1)*S2/S3
            IF DIST<.4*DELH/N1 THEN PSET(I,J),14
            GOTO repeat
         END IF
            X=X1 : Y=Y1 : U=U1 : V=V1
      NEXT K
repeat:
   NEXT J : NEXT I
END
```

Met hetzelfde programma kan ook de omtrek van M in zijn geheel worden gecreëerd, het resultaat is reeds afgebeeld in figuur 10.1. De naam van het enigszins aangepaste programma luidt MANDISTM.

```
REM ***Mandelbrot set met de afstandsfunctie***
REM ***totaalbeeld monochroom***
REM ***naam:MANDISTM***
   DEFDBL A-Z
   SCREEN 12 : CLS
   XM=320 : YM=240
   AC=-.65 : DELH=1.35 : DELV=1.1
   N1=200 : N2=INT(N1*DELV/DELH)
   FOR I=-N1 TO N1 : FOR J=0 TO N2
      A=AC+I*DELH/N1 : B=J*DELV/N2
      S1=4*(A*A+B*B) : S2=S1-2*A+1/4
      IF S1+8*A+15/4<0 THEN COL=14 : GOTO graphics
      IF S2-SQR(S2)+2*A-1/2<0 THEN COL=14 : GOTO graphics
      X=A : Y=B : U=1 : V=0
```

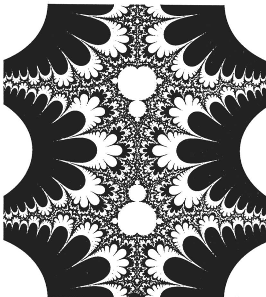
```
            DIS=SQR(W1)*LOG(W1)/SQR(U1*U1+V1*V1)
              IF W1>64 THEN
                  IF DIS>.4*DELH/N1 THEN COL=0 ELSE COL=14
                  GOTO graphics
              END IF
          NEXT K : COL=14
graphics:
          PSET (XM+I,YM-J),COL : PSET (XM+I,YM+J),COL
      NEXT J : NEXT I
END
```

10.5 Nog meer Mandelbrot sets

De Mandelbrot set is in feite een soort atlas van een familie van Julia sets. Hieruit volgt dat op analoge wijze een andere familie van Julia sets zoals bepaald door $z \to f(z) + c$, waarin $f(z)$ een bepaalde functie van z is, eveneens een soort Mandelbrot set moet opleveren. We kunnen bijvoorbeeld voor $f(z)$ een hogere veelterm zoals z^2 of z^3 nemen. In het programma MANDELX2 is $f(z) = \pi \sin(z)$ gekozen. Om de wiskundige inhoud van het programma te begrijpen is meer dan de gebruikelijke kennis van wiskunde nodig. Daarom wordt er niet op ingegaan, maar wordt volstaan met de mededeling dat de structuur niet afwijkt van die van MANDELX1, het standaard programma van de traditionele Mandelbrot set. De toets op grond waarvan bepaald wordt of een punt c tot de Mandelbrot set behoort of niet, heeft in het algemeen betrekking op de baan van een kritiek punt van $f(z)$, een punt waarvoor de afgeleide van $f(z)$ verdwijnt. De afgeleide van de sinus is de cosinus, en omdat geldt $\cos(\pi/2) = 0$, moeten we de baan van $\pi/2$ nemen. Het resultaat van MANDELX2 is afgebeeld in figuur 10.5.

```
REM ***Mandelbrot set van c+pi*sin(z) ***
REM ***naam:MANDELX2***
    DEFDBL A-Z
    SCREEN 12 : CLS : PI=4*ATN(1)
    WINDOW (-320,-240)-(319,239)
REM ***kleuren***
    DIM COL(8) : DATA 0,1,9,2,10,4,12,14,6
    FOR I=0 TO 8 : READ COL(I) : NEXT I
    DELH=PI : DELV=.75*PI  'grootte Mset
    N1=240 : N2=INT(N1*DELV/DELH)
    FOR I=0 TO N1 : FOR J=0 TO N2
        IF INKEY$<>"" THEN END
        A=DELH*I/N1 : B=DELV*J/N2
        X=PI/2 : Y=0
        FOR K=1 TO 64
            IF ABS(Y)>12 THEN L=1+K MOD 8 : GOTO graphics
```

```
        U=EXP(Y) : V=1/U : CH=(U+V)/2 : SH=(U-V)/2
        SS=SIN(X) : CS=COS(X)
        X1=A+PI*SS*CH : Y1=B+PI*CS*SH
        DIST=ABS(X-X1)+ABS(Y-Y1)
        IF DIST<.001 THEN L=3 : GOTO graphics
        X=X1 : Y=Y1
      NEXT K : L=3
graphics:
      PSET (I,J),COL(L) : PSET (I,-J),COL(L)
      PSET (-I,J),COL(L) : PSET (-I,-J),COL(L)
    NEXT J : NEXT I
END
```



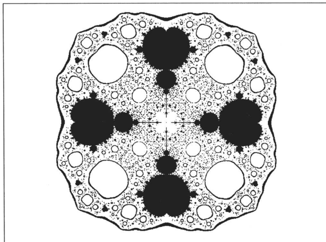
Figuur 10.5 Een Mandelbrot set van een sinus-functie

Als tweede voorbeeld van een ander soort Mandelbrot set gaan we uit van het iteratieve systeem:

$z \rightarrow c\,(z^2 + z^{-2})$

waarin c een willekeurige complexe constante is.

Indachtig de zojuist gemaakt opmerking moeten we testen op de baan van de z-waarde waarvoor de afgeleide van het rechterlid nul is, en die is $z = 1$. Een theoretische beschouwing toont aan dat de daardoor bepaalde Mandelbrot set een viervoudige symmetrie heeft. Een monochrome versie van het programma MANDELX3, afgebeeld in figuur 10.6, laat de globale vorm van de bekende Mandelbrot set zien als een soort klavertje vier. Verder bevat de figuur nog talloze overeenkomstige miniatuurtjes.



Figuur 10.6 Mandelbrot kwartet

```
REM ***Mandelbrot set van z'=c*(z*z+1/z*z)***
REM ***naam:MANDELX3***
   SCREEN 12 : CLS
   DEFDBL A,B,D,X,Y,S
   WINDOW (-320,-240)-(319,239)
   DELH=.8 : DELV=.8
   N1=200 : N2=INT(N1*DELV/DELH)
REM ***hoofdprogramma***
   FOR I=0 TO N1 : A=I*DELH/N1
      FOR J=0 TO N2 : B=J*DELV/N2
         X=1 : Y=0
         FOR K=1 TO 100
            S=X*X+Y*Y : S2=S*S+1E-8
            IF S>64 THEN L=1+K MOD 14 : GOTO graphics
```

```
            X2=X*X-Y*Y  :  Y2=2*X*Y
            XR=X2/S2  :  YR=-Y2/S2
            X1=X2+XR  :  Y1=Y2+YR
            X=A*X1-B*Y1  :  Y=A*Y1+B*X1
        NEXT K  :  L=9
graphics:
        PSET (I,J),L  :  PSET (I,-J),L
        PSET (-I,J),L  :  PSET (-I,-J),L
        IF INKEY$<>"" THEN END
      NEXT J
   NEXT I
END
```

10.6 Een zoom-programma maken

In deze paragraaf wordt een interactief programma MZOOML ontworpen waarbij tijdens het maken van een detail van de Mandelbrot set via het toetsenbord ingegrepen kan worden. De bedoeling daarvan is onder meer om uit het reeds verkregen plaatje een interessant kleiner detail te selecteren voor een verdere bewerking. Dergelijke interrupts zijn op verschillende manieren mogelijk. Hier is gekozen door ergens in het lopende programma, de hoofdlus, een INKEY$-opdracht te plaatsen en de daardoor gegenereerde scancode te gebruiken (zie bijvoorbeeld QBW, appendix C of QBG, pagina 261). Het raamwerk van het programma MZOOML is:

```
voorbereidende administratie
startdata
DO
   bepaling kleurelementen
   A$=INKEY$
   IF LEN(A$)>1 THEN GOSUB keycontrol
LOOP
END
```

De subroutine KEYCONTROL kunnen we beschouwen als een multiswitch die voor een aantal functietoetsen en de pijltjestoetsen bepaalt welke acties er uitgevoerd moeten worden. De genoemde subroutine is een SELECT CASE-constructie waarin de door MID$(A$,2,1) bepaalde toetswaarde wordt geëvalueerd. Zo wordt een detail geselecteerd met een rechthoek waarvan de grootte en de plaats met behulp van F5, F6 en de pijltjestoetsen interactief geregeld kunnen worden.

Een dergelijk programma heeft alleen zin als het niet te veel tijd kost. Daarom verloopt de berekening – of liever de kleuring van M – in een aantal fasen met een van heel grof tot fijn verlopende resolutie. In de

eerste fase wordt het beeld heel globaal opgebouwd uit een beperkt aantal vierkantjes, 16 horizontaal en 14 verticaal in blokken van 32*32 pixels. In de volgende fasen wordt het beeld opnieuw opgebouwd met vierkantjes van, gerekend in de lineaire maat, telkens een halve grootte, dus 16*16, 8*8, 4*4, 2*2 en ten slotte een enkele pixel. Weliswaar worden daarbij sommige pixels meermalen berekend, maar het daardoor ontstane tijdverlies bedraagt bij een geheel voltooid beeld toch slechts zo'n 30%.

De beeldvorming gaat vooral in de eerste fasen razend snel en het is prettig wanneer we meteen al een selectie kunnen maken voor een kleiner detail zonder dat we moeten wachten tot het beeld voltooid is. Het is handig wanneer we bij het maken en selecteren van details over wat numerieke informatie kunnen beschikken en tijdens de programma-uitvoering een andere waarde aan een parameter kunnen toewijzen. Dat kan via het keycontrol-mechanisme allemaal gemakkelijk geregeld worden. Een en ander maakt het wenselijk het beeldscherm te verdelen in een tekstgedeelte en een grafisch gedeelte. In het programma wordt dat met de VIEW-opdracht bereikt, waarbij voor het grafisch gedeelte een rechthoek in de verhouding 8 horizontaal, 7 verticaal wordt gereserveerd. Aan de bovenkant is een smalle strook gereserveerd voor een paar getallen en links een verticale strook voor tekst en enkele numerieke gegevens.

De selectierechthoek wordt bepaald door de coördinaten (XA,YA) en (XB,YB) van de hoekpunten A linksboven en B rechtsonder. In de neutrale stand valt de rechthoek nagenoeg samen met het grafische venster (128,32)-(640,480). De zijden van de rechthoek zijn opgebouwd uit puntjes, 40 horizontaal bij 35 verticaal. Met de interrupttoetsen kan de rechthoek bestuurd worden en daarbij laten we de waarden van XA,XB in stapjes van acht pixels, en die van YA,YB in stapjes van zeven pixels veranderen. De selectierechthoek behoudt dus steeds zijn vorm van 8:7 en het aantal puntjes op de omtrek blijft gelijk. Naarmate de rechthoek kleiner wordt, liggen de puntjes dichter tegen elkaar aan, hetgeen de zichtbaarheid bevordert. Voor de technische programmering is wel enige arbeid nodig. In elk geval moeten bij de beweging de kleurattributen van de pixels die door de selectierechthoek overschreven worden, bewaard worden om ze later in de juiste kleur te kunnen herstellen. De kleurattributen worden opgeslagen in de arrays CU, CD, CL en CR. Het bijbehorende programmafragment is ondergebracht in de subroutine PIXELCONTROL.

Het programma bevat nog een aantal details. Sommige daarvan zijn niet strikt noodzakelijk, maar maken het geheel wat gebruiksvriendelijker. Een nuttige faciliteit is de verhoging of verlaging van NMAX, het aantal

malen dat de hoofdlus voor de berekening van de kleurwaarde van een pixel doorlopen moet worden. Die waarde wordt weliswaar enigszins automatisch bijgesteld, maar de kans bestaat dat sommige delen van het scherm alleen maar zwart blijven omdat die waarde te laag is. Anderzijds geeft een hoge waarde van NMAX misschien onnodig tijdverlies. Bij de stap voor stap vergroting van een detail kan men hoge waarden halen. In het grafische gedeelte worden altijd de coördinaten van het middelpunt van het beeld en de halve horizontale breedte vermeld. Die gegevens, eventueel tezamen met NMAX, leggen het detail voldoende vast, en kunnen desgewenst worden gebruikt bij een ander programma zoals MANDET of MANDIST. De mogelijkheden van het rekenen met dubbele precisie zijn echter begrensd en men kan dus niet voortdurend doorgaan met het vergroten van hetzelfde detail. Gaat men te ver, dan verraadt zich dit in het optreden van grotere kleurvlakken waar men dit niet verwacht. Het volledige programma luidt als volgt:

```
REM ***Mandelbrot zooming tiling version ***
REM ***name:MZOOML***
  SCREEN 12 : CLS
  DEFDBL A,B,D,S-Z : DEFINT C,F,I-N
  DIM CU(40),CD(40),CL(35),CR(35)
  XM=384 : YM=256 : NFIXED=50 : NBEGIN=50
  GOSUB startdata : CLS
  IF A$="X" OR A$="x" THEN END
  DO
    F2=0 : F7=0 : F10=0  'switches for text, detail and end
    XA=128 : YA=32 : XB=640 : YB=480 : ACT=AC : BCT=BC
    VIEW SCREEN (128,32)-(638,478),,15
    GOSUB textonoff : GOSUB printdata
    FOR KP=0 TO 5
      FP1=16*2^KP-1 : FP2=14*2^KP-1 : FP3=2^(5-KP)
      FOR JP=0 TO FP2 :  YP=32+JP*FP3
   FOR IP=0 TO FP1 :  XP=128+IP*FP3
REM ***calculation of pixelvalue***
     A=ACT+(XP-XM)*DELH/256 : B=BCT-(YP-YM)*DELV/224
     U=4*(A*A+B*B) : V=U-2*A+1/4 : COL=0
     IF U+8*A+15/4>0 AND V-SQR(V)+2*A-1/2>0 THEN
       X=A : Y=B
       FOR K=1 TO NMAX
         Z=X : X=X*X-Y*Y+A : Y=2*Z*Y+B
         S=X*X+Y*Y : IF S>128 THEN EXIT FOR
       NEXT K
       IF K<INT(NMAX/2) THEN COL=1+K MOD 15
     END IF
     IF KP=5 THEN
       PSET (XP,YP),COL  'print pixel
     ELSE
       LINE (XP,YP)-(XP+FP3-1,YP+FP3-1),COL,BF  'square
```

```
      END IF
      A$=INKEY$
      IF LEN(A$)>1 THEN GOSUB keycontrol : IF F7 THEN EXIT
FOR
    NEXT IP : IF F7 THEN EXIT FOR
        NEXT JP : IF F7 THEN EXIT FOR
        LINE (129,32)-(130,479),15,BF
    NEXT KP
    IF F7=0 THEN
      DO : A$=INKEY$ : IF LEN(A$)>1 THEN GOSUB keycontrol
      LOOP UNTIL F7
    END IF
  LOOP UNTIL F10
  SCREEN 0 : PRINT"Press any key";
END
startdata:
  PRINT"Mandelbrot Set"
  PRINT"=============="
  PRINT"Give the X coordinate centre (default value -.75): ";
  INPUT"X=",AC : PRINT
  PRINT"     the Y coordinate centre (default value 0): ";
  INPUT"    Y="",BC : PRINT
  PRINT"     halfwidth D (default value 1.6): ";TAB(52);
  INPUT"D=",DELH
  IF AC=0 AND BC=0 AND DELH=0 THEN AC=-.75 : DELH=1.6
  DELV=7*DELH/8
  NMAX=INT(LOG(2/DELH)*NBEGIN)
  IF NMAX<NBEGIN THEN NMAX=NBEGIN
  PRINT : PRINT "Press 'X' to eXit the program"
  PRINT "Press any key to start the program"
  DO : A$=INKEY$ : LOOP UNTIL A$<>""
RETURN
keycontrol:
  CS=ASC(MID$(A$,2,1))
  SELECT CASE CS
    CASE 60 : GOSUB textonoff                            'F2
    CASE 61,62 : GOSUB newmax                            'F3,F4
    CASE 63,64 : GOSUB zoom                              'F5,F6
    CASE 65 : F7=1 : GOSUB newdata                       'F7
    CASE 67 : RUN                                         'F9
    CASE 68 : F7=1 : F10=1                               'F10
    CASE 72,75,77,80 : GOSUB moveframe                   'arrows
  END SELECT
RETURN
textonoff:
   F2=(F2+1) MOD 2
   FOR I=1 TO 24 : LOCATE I,1 : PRINT SPACE$(14); : NEXT I
   LOCATE 1,1 : PRINT SPACE$(80)
   IF F2 THEN GOSUB printdata : GOSUB helptext
```

```
   LOCATE 1,1
RETURN
printdata:
   LOCATE 1,1 : PRINT"Current DATA"
   LOCATE 1,20 : PRINT"AC = "; : PRINT AC
   LOCATE 1,50 : PRINT"BC = "; : PRINT BC
   LOCATE 3,1 :  PRINT"D = "; : PRINT USING"||.||^^^^"";DELH
   DELV=7*DELH/8 : PRINT
   LOCATE 5,1 : PRINT"MAX=";NMAX
RETURN
helptext:
   LOCATE 7,1 : PRINT"Keys:" : PRINT
   PRINT"F2 text on/off"
   PRINT"F3 MAX down"
   PRINT"F4 MAX up"
   PRINT"F5 zoom IN"
   PRINT"F6 zoom OUT" : PRINT
   PRINT"zoom and" : PRINT"use arrows" : PRINT
   PRINT"F7 show detail" : PRINT
   PRINT"F9 restart" : PRINT
   PRINT"F10 end"
RETURN
zoom:
  IF CS=63 THEN
    XAN=XA+8 : XBN=XB-8 : YAN=YA+7 : YBN=YB-7
    IF XBN-XAN<12 THEN XAN=XA : XBN=XB : YAN=YA : YBN=YB
  END IF
  IF CS=64 THEN XAN=XA-8 : XBN=XB+8 : YAN=YA-7 : YBN=YB+7
  GOSUB pixelcontrol
RETURN
newdata:
  IF newframe=0 THEN RETURN
  AC=ACT+((XA+XB)/2-XM)*DELH/256
  BC=BCT-((YA+YB)/2-YM)*DELV/224
  DELH=DELH*(XB-XA)/512 : DELV=7*DELH/8
newmax:
  IF CS=61 THEN DF=1.1 ELSE DF=1/1.1
  NBEGIN=INT(DF*NBEGIN)
  NMAX=INT(LOG(2/DELH)*NBEGIN)
  IF NMAX>5000 THEN
    NMAX=5000
    DO
      NBEGIN=INT(NBEGIN/1.1)
    LOOP UNTIL INT(LOG(2/DELH)*NBEGIN)<=5000
  END IF
  IF NMAX<NFIXED THEN
    NMAX=NFIXED
    DO
      NBEGIN=INT(NBEGIN*1.1)
```

```
    LOOP UNTIL INT(LOG(2/DELH)*NBEGIN)>=NFIXED
  END IF
  GOSUB printdata
  newframe=0
RETURN
moveframe:
  IF CS=72 THEN XAN=XA : XBN=XB : YAN=YA-7 : YBN=YB-7
  IF CS=75 THEN XAN=XA-8 : XBN=XB-8 : YAN=YA : YBN=YB
  IF CS=77 THEN XAN=XA+8 : XBN=XB+8 : YAN=YA : YBN=YB
  IF CS=80 THEN XAN=XA : XBN=XB : YAN=YA+7 : YBN=YB+7
  GOSUB pixelcontrol
RETURN
pixelcontrol:
  FOR I=0 TO 40 : X=((40-I)*XA+I*XB)/40  'restore old frame
    PSET (X,YA),CU(I) : PSET (X,YB),CD(I)
  NEXT I
  FOR J=1 TO 35 : Y=((35-J)*YA+J*YB)/35
    PSET (XA,Y),CL(J) : PSET (XB,Y),CR(J)
  NEXT J
  FOR I=0 TO 40 : X=((40-I)*XAN+I*XBN)/40  'save new frame
    CU(I)=POINT (X,YAN) : CD(I)=POINT (X,YBN)
  NEXT I
  FOR J=1 TO 35 : Y=((35-J)*YAN+J*YBN)/35
    CL(J)=POINT (XAN,Y) : CR(J)=POINT (XBN,Y)
  NEXT J
  XA=XAN : YA=YAN : XB=XBN : YB=YBN
  newframe=1  'set new frame
  FOR I=0 TO 40 : X=((40-I)*XA+I*XB)/40
    PSET (X,YA),15 : PSET (X,YB),15
  NEXT I
  FOR J=1 TO 35 : Y=((35-J)*YA+J*YB)/35
    PSET (XA,Y),15 : PSET (XB,Y),15
  NEXT J
RETURN
```

Literatuur

Aitken, P., *QuickBasic 4.5; programmeren voor gevorderden.* Academic Service, 1990

Barnsley, M.F., *Fractals everywhere.* Academic Press, 1988

Broer, H.W. en F. Verhulst, *Dynamische Systemen en Chaos; een revolutie vanuit de wiskunde.* Epsilon Uitgaven, dl.14. Utrecht, 1990

Feldman, Ph. en T. Rugg, *Werken met QBasic.* Academic Service, 1991

Gleick, J., *Chaos. De derde wetenschappelijke revolutie.* Contact, 1989

Lauwerier, H.A., *Analyse met de microcomputer.* Epsilon uitgaven dl.7, tweede druk. Utrecht, 1989

Lauwerier, H.A., *Meetkunde met de microcomputer.* Epsilon uitgaven, dl.8. Utrecht, 1987

Lauwerier, H.A., *Modellen met de microcomputer; experimentele wiskunde.* Epsilon Uitgaven, dl.12, tweede druk. Utrecht, 1990

Lauwerier, H.A., *Fractals, meetkundige figuren in eindeloze herhaling.* Vijfde druk. Aramith, 1992

Lauwerier, H.A., *Een wereld van fractals.* Aramith, 1990

Lauwerier, H.A., *Computer simulaties; de wereld als model.* Aramith, 1992

Mandelbrot, B.B., *The fractal geometry of nature.* Freeman and Co., 1977

Stewart, I., *Speelt god een spelletje, de structuur van de chaos.* Spectrum, 1991

Veen, N. van, H. Bakker en G. Bruijnes, *Basiscursus QBASIC.* Academic Service, 1993

Programma's per hoofdstuk

Hoofdstuk 1
- EGACOL1
- EGACOL2
- MIXER

Hoofdstuk 2
- ASCPRINT
- ASCRIJ
- WOORD
- KATTEN
- CIRKELB
- FOURIER
- LISSA
- LISSAX
- LISSAV
- ANIMA
- STARMOVE

Hoofdstuk 3
- WEB
- KANT
- STERPQ
- ASTROIDE
- CYCLO1
- CYCLO2
- TURNLINE
- WERVEL
- OMHUL
- CAUSTIC
- KALEIDO

Hoofdstuk 4
- PATROON1
- PATROON2
- PATROON3
- PATROON4
- ARTBLOK
- ARTBLOKQ
- STEMPEL
- TRUCHET

Hoofdstuk 5
- FRACMC1
- FRACMC2
- FRACMC3
- FRACMC4
- FRACMC5
- FRACMC6
- FRACMC7
- BLAD
- VAREN
- SQUAREF

Hoofdstuk 6
- GROEIM1
- GROEIM2
- HENON1
- HENON2
- DYNSYSX1
- DYNSYSX2
- MIRADSX1
- MIRADSX2
- MIRAMUS

Hoofdstuk 7
- CONWAY1
- CONWAYR
- CONWAYS
- CONWAY2

Hoofdstuk 8

- TURTLE
- TURTLEK
- TURTLEK1
- TURTLEK2
- TURTLEM
- TURTLEK3
- TURTLEK4
- TURTLES
- TURTLEA

Hoofdstuk 9

- JULIAMC
- JULFILL
- JULFILLX
- JULBSC
- JULDIST
- JULIAP
- JULIAPX
- JULIAT

Hoofdstuk 10

- MANDELX1
- MANDET
- MANDIST
- MANDISTM
- MANDELX2
- MANDELX3
- MZOOML

176

Index

G

H

I

J

K

L

M

O

P



Spelen met de computer. Nu eens niet met door anderen bedachte computerspellen, maar met zelfgemaakte programma's waarmee je spelenderwijs de mooiste graphics op het beeldscherm kunt toveren. Natuurlijk is het leuk om meteen achter de computer te gaan zitten en de programma's, die ook op de meegeleverde diskette staan, in dit boek te draaien. Nog leuker is het om de programma's een beetje te wijzigen, andere getallen te proberen of met de programma's als voorbeeld zelf iets nieuws te ontwerpen. Als je enige kennis hebt van BASIC en niet schrikt van een eenvoudige wiskundige formule, kun je direct aan de slag. De programma's in dit boek zijn enigszins geordend van makkelijk naar moeilijk. Spelenderwijs doe je ervaring op met het werken met grafische programma's. Beginnende computeraars kunnen een letter of een woord afbeelden of een patroon voor een borduurkleedje ontwerpen.

Met wat ervaring kan je in de laatste hoofdstukken mooie, naar Julia en Mandelbrot genoemde plaatjes maken. Verder zijn er fractals en vreemde verrassingen. Ook is er het een en ander over chaos. Daarnaast zijn er enkele programma's over een echt spel, het beroemde 'game of life' van Conway. Kortom, dit is een boek om mee bezig te zijn, om al spelende wijzer te worden en inzicht te krijgen in moderne begrippen als chaos en fractals. Maar ook een boek waarmee je kunt genieten van prachtige plaatje die je met een moderne pc en wat programmeervernuft kunt maken.

Prof. dr. H. A. Lauwerier is emeritus hoogleraar in de zuivere en toegepaste wiskunde aan de Universiteit van Amsterdam en was als hoofd van de afdeling toegepaste wiskunde verbonden aan het mathematisch centrum in Amsterdam (CWI).

ACADEMIC SERVICE
informatica

NUGI 852

ISBN 90 395 0092 4

Met gratis programmadiskette

